#5

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: G. Kondo

Serial No.: 10/064,751

Filed: August 13, 2002

For: APPLICATION EDITING APPARATUS AND DATA PROCESSING METHOD AND PROGRAM

Date: October 21, 2002

Docket No.: JP920000461US1

Group Art Unit: 2123

Assistant Commissioner for Patents
Washington, D.C. 20231

SUBMISSION OF PRIORITY DOCUMENT

Sir:

Enclosed herewith is a certified copy of Japanese Application No. 2001-246290 filed August 14, 2001, in support of applicant's claim to priority under 35 U.S.C. 119.

Respectfully submitted,

Derek S. Jennings
Reg. Patent Agent/Patent Engineer
Reg. No. 41,473
(914) 945-2144

IBM CORPORATION
Intellectual Property Law Dept.
P. O. Box 218
Yorktown Heights, N. Y. 10598

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　８月１４日

出願番号
Application Number: 特願２００１－２４６２９０

[ST.10/C]: [ＪＰ２００１－２４６２９０]

出願人
Applicant(s): インターナショナル・ビジネス・マシーンズ・コーポレーション

２００２年　３月１５日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００２－３０１７５５４

【書類名】　特許願
【整理番号】　JP9000461
【提出日】　平成13年 8月14日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F　7/00
【発明者】
　【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内
　【氏名】　近藤　豪
【特許出願人】
　【識別番号】　390009531
　【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション
【代理人】
　【識別番号】　100086243
　【弁理士】
　【氏名又は名称】　坂口　博
【代理人】
　【識別番号】　100091568
　【弁理士】
　【氏名又は名称】　市位　嘉宏
【代理人】
　【識別番号】　100106699
　【弁理士】
　【氏名又は名称】　渡部　弘道
【復代理人】
　【識別番号】　100104880
　【弁理士】
　【氏名又は名称】　古部　次郎

【選任した復代理人】

【識別番号】　100100077

【弁理士】

【氏名又は名称】　大場　充

【手数料の表示】

【予納台帳番号】　081504

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【包括委任状番号】　0004480

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　アプリケーション編集装置、データ処理方法及びプログラム

【特許請求の範囲】

【請求項１】　コンピュータを用いて、モデルとビューとを別個に有するアプリケーションの編集を行うアプリケーション編集装置において、

前記アプリケーションにおける第１のモデルを編集する編集手段と、

前記編集手段にて編集された第１のモデルを第２のモデルに変換するモデル変換手段と、

前記第２のモデルを当該第２のモデルのビューにてディスプレイ装置に表示するビュー表示手段とを備え、

前記ビュー表示手段は、

前記編集手段による前記第１のモデルの編集に基づいて前記第２のモデルが更新された場合に、当該第２のモデルの更新に基づくイベントを生成するイベント生成手段を備え、

前記イベント生成手段にて生成されたイベントに基づいて前記ディスプレイ装置に表示されたビューを変更することを特徴とするアプリケーション編集装置。

【請求項２】　前記ビュー表示手段は、前記編集手段による前記第１のモデルの編集に基づいて前記第２のモデルが更新された場合に、当該第２のモデルの更新前後の差分を抽出する差分抽出手段をさらに備え、

前記イベント生成手段は、前記差分抽出手段にて抽出された前記差分に関する情報をパラメータとして前記イベントを生成することを特徴とする請求項１に記載のアプリケーション編集装置。

【請求項３】　前記モデル変換手段は、前記第１のモデルを要素ごとに前記第２のモデルの対応要素に変換することを特徴とする請求項１に記載のアプリケーション編集装置。

【請求項４】　前記モデル変換手段は、前記第１のモデルにおける要素の変換結果に対応する要素が前記第２のモデルに存在しない場合に、当該変換結果に対応する要素を当該第２のモデルに付与することを特徴とする請求項１に記載の

アプリケーション編集装置。

【請求項５】　前記モデル変換手段は、前記第１のモデルにおける前記編集手段により編集された要素を、前記第２のモデルにおける対応する要素に変換し、変換された当該要素を用いて前記第２のモデルを更新することを特徴とする請求項１に記載のアプリケーション編集装置。

【請求項６】　コンピュータを用いて、モデルとビューとを別個に有するアプリケーションの編集を行うアプリケーション編集装置において、

前記アプリケーションにおける第１のモデルを編集する編集手段と、

前記編集手段にて編集された第１のモデルを第２のモデルに変換するモデル変換手段と、

前記第２のモデルを当該第２のモデルのビューにてディスプレイ装置に表示するビュー表示手段と、

前記第１のモデルの更新を当該第１のモデルのビューに反映させるためのイベントを、前記第１のモデルを前記第２のモデルに変換するための変換ルールを用いて、前記第２のモデルのビューを変更するイベントに変換するイベント変換手段とを備え、

前記ビュー表示手段は、前記イベント変換手段にて生成されたイベントに基づいて、前記ディスプレイ装置に表示されたビューを変更することを特徴とするアプリケーション編集装置。

【請求項７】　コンピュータを用いて、所定のアプリケーションにおけるモデルを他のアプリケーションにおけるビューにて表示するデータ処理方法において、

前記所定のアプリケーションにおける第１のモデルが更新された場合に、前記アプリケーションを格納したデータ記憶手段から前記他のアプリケーションにおける第２のモデルを読み出し、当該更新内容を反映させて当該第２のモデルを更新するステップと、

前記第２のモデルの更新に基づくイベントを生成し、当該イベントに基づいてディスプレイ装置に表示された前記他のアプリケーションにおけるビューを変更するステップと

を含むことを特徴とするデータ処理方法。

【請求項８】　前記他のアプリケーションにおけるビューを変更するステップは、

前記第２のモデルにおける更新前後の差分を抽出するステップと、

更新前の前記第２のモデルに前記差分に相当する変更を加えることにより前記イベントを生成するステップと、

当該イベントに基づいて前記ビューを変更するステップと

を含むことを特徴とする請求項７に記載のデータ処理方法。

【請求項９】　前記第２のモデルを更新するステップは、前記第１のモデルを要素ごとに前記第２のモデルの対応要素に変換するステップを含み、

前記他のアプリケーションにおけるビューを変更するステップは、前記第２のモデルにおける変換された要素ごとに当該第２のモデルにおける更新前後の差分を抽出するステップを含むことを特徴とする請求項８に記載のデータ処理方法。

【請求項１０】　前記他のアプリケーションにおけるビューを変更するステップは、

前記第１のモデルの更新を前記所定のアプリケーションにおけるビューに反映させるためのイベントを、前記第１のモデルを前記第２のモデルに変換するための変換ルールを用いて、前記他のアプリケーションにおけるビューを変更するイベントに変換するステップを含むことを特徴とする請求項７に記載のデータ処理方法。

【請求項１１】　コンピュータを制御して、モデルとビューとを別個に有するアプリケーションを実行するプログラムにおいて、

前記アプリケーションを格納したデータ記憶手段から前記アプリケーションにおけるモデルを読み出し、当該モデルのビューをディスプレイ装置に表示する処理と、

前記モデルが更新された場合に、当該モデルの更新前後の差分を抽出する処理と、

抽出された前記差分に基づいて前記ビューを変更するためのイベントを生成する処理と、

生成された前記イベントに基づいて前記ディスプレイ装置に表示されている前記ビューを変更する処理と

を前記コンピュータに実行させることを特徴とするプログラム。

【請求項１２】　コンピュータを制御して、モデルとビューとを別個に有するアプリケーションを編集するプログラムにおいて、

前記アプリケーションにおける第１のモデルに対する編集を行う編集手段と、

前記編集手段にて編集された第１のモデルを第２のモデルに変換するモデル変換手段と、

前記モデル変換手段により前記第１のモデルが前記第２のモデルに変換された場合に、当該第２のモデルと直前に変換された前記第２のモデルとの差分を抽出する差分抽出手段と、

前記差分抽出手段にて抽出された差分に基づくイベントを生成するイベント生成手段と、

前記第２のモデルを当該第２のモデルのビューにてディスプレイ装置に表示すると共に、前記イベント生成手段にて生成されたイベントに基づいて前記ディスプレイ装置に表示されたビューを変更するビュー表示手段として、

前記コンピュータを動作させることを特徴とするプログラム。

【請求項１３】　前記モデル変換手段は、前記第１のモデルを要素ごとに前記第２のモデルの対応要素に変換することを特徴とする請求項１２に記載のプログラム。

【請求項１４】　前記差分抽出手段は、前記第２のモデルにおける変換された要素ごとに当該第２のモデルにおける更新前後の差分を抽出することを特徴とする請求項１２に記載のプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、モデルオブジェクトとビューオブジェクトとを別個に有するアプリケーションにおけるモデル変換及びビューの表示方法に関する。

【０００２】

【従来の技術】

アプリケーションプログラム（以下、アプリケーション）を対話的に作成する際の基本的な手法として、アプリケーションのロジックであるモデルオブジェクト（以下、モデル）と、ディスプレイ装置に対する当該アプリケーションの表示態様であるビューオブジェクト（以下、ビュー）とに分割する方法がある。アプリケーションをモデルとビューとに分割することにより、１つのモデルに対して、種々のビューを対応づけることができる。すなわち、１つのプログラムを処理や操作に応じた種々の表示態様で利用することが可能となる。

【０００３】

図３６は、モデルとビューとの関係を例示する図である。

図３６に示す例では、１つのモデル（ツリー構造）に対してWysiwyg表示、ツリー表示、ソース表示の３種類のビューが与えられている。すなわち、ユーザは、図示のモデルで特定されるアプリケーションを、閲覧や編集といった用途に応じてＷＹＳＩＷＹＧ（What You See Is What You Get）表示、ツリー表示、ソース表示の中から任意のビューを選んで表示することができる。

【０００４】

このようなアプリケーションでは、所定のビューを通してモデルの変更が加えられた場合、その変更が他のビューに対しても反映される。

図３７は、図３６のアプリケーションにおいて、所定のビューにおける変更がモデル及び他のビューに反映する様子を示す図である。

図３７において、ＷＹＳＩＷＹＧ表示のビューに書き込みが行われると、次の手順で他のビューに反映される（手順の数字は図中の数字に対応）。

１．ビューに書き込みが発生。

２．変更が発生したビューオブジェクトに対応するモデルオブジェクトを変更（図示の例ではノードの追加）。

３．変更内容に対応するイベントを生成して各ビューに同報。

４．各ビューが受け取ったイベントを解釈して更新。

【０００５】

ところで、上記のようにモデルとビューとを分割したアプリケーションでは、

１つのモデルに対して複数のビューを自由に追加削除できるという利点がある。ただし、所定のモデルに対して所定のビューを使用するためには、モデルの内容に対応した表示を行ったり、モデルに変更があった場合に生成される上記のイベントを適切に処理して反映させたりするために、当該ビューは、当該モデルに対応した一定のインターフェイスに合致させたものである必要がある。

所定のモデルに対し所定のビューを追加しようとする場合で、かつ当該モデルに対応したインターフェイスを持ったビューが存在しない場合を考える。これを解決するために、幾つかの方法が考え得る。最も直接的な方法は、要求されるインターフェイスを持った必要なビューを作成する方法である。ただし、所望のモデルに対して所望のビューをいちいち作成するとすれば、作業コストがかかりすぎるという問題がある。

【０００６】

そこで、当該モデルにおいて要求されるインターフェイスとは異なる、他のアプリケーションで使用されるインターフェイスを持ったビューが存在する場合、このビューを当該モデル用に使用することが考えられる。Adapterパターンとして知られる手法などを使ってインターフェイスを適応させることにより、当該ビューを所望のモデルに対して使用することができる。Adapterパターンについては、例えば次の文献に詳細に記述されている。

文献：Gamma, E., Helm, R., Johnson , R., Vlissides, J. Design Patterns, 1995.

【０００７】

図３８は、Adapterパターンを模式的に例示する図である。

図３８（Ａ）に示すように、アプリケーションＡとアプリケーションＢが、それぞれ独自のインターフェイスに基づいたモデルとビューの対を持っているものとする。この場合、アプリケーションＢのモデルＢを表示するためにアプリケーションＡのビューＡを使用することはできない。しかしながら、図３８（Ｂ）に示すように、アプリケーションＡ、Ｂのインターフェイスを適合させるAdapter

を介在させることにより、モデルＢを表示するためにビューＡを用いることが可能となる。

【０００８】

また、他のアプリケーションで使用されるビューを使用するための他の方法として、モデル変換器を用いて所望のモデルの内容を当該他のアプリケーションにおけるモデルに反映させ、当該他のアプリケーションにおけるビューで表示する方法が考えられる。

図３９は、モデル変換により他のアプリケーションのビューを用いて表示を行う方法を説明する図である。

図３９に示すように、アプリケーションＡにはモデルＡを表示するためにＷＹＳＩＷＹＧ表示のビューＡ１及びツリー表示のビューＡ２があり、アプリケーションＢにはモデルＢを表示するためにソース表示のビューＢ１及びツリー表示のビューＢ２があり、アプリケーションＣにはモデルＣを表示するためにソース表示のビューＣがある。ここで、アプリケーションＡにおいて、ソース表示を行いたい場合、アプリケーションＡにはソース表示を行うビューがない。そこで、モデルＡを、モデル変換器を用いてモデルＢまたはモデルＣに変換し、ビューＢ１またはビューＣを用いてソース表示を行う。同様に、アプリケーションＢやアプリケーションＣでＷＹＳＩＷＹＧ表示を行いたい場合や、アプリケーションＣでツリー表示を行いたい場合、それぞれのビューを持ったアプリケーションのノードに変換することによって、当該変換先のアプリケーションが持つビューを利用することができる。

【０００９】

上記のモデル変換器の代表的な例としてＸＳＬＴ（XSL Transformations）プロセッサがある。ＸＳＬＴにおいて、モデル変換を行うためのルールは、ＸＳＬ（eXtensible Style Language）で記述される。

ＸＳＬＴプロセッサの代表的な実装に、Ａｐａｃｈｅプロジェクトから提供されているＸａｌａｎがある。Ｘａｌａｎは、ストリーム以外にＸＭＬにおける上記モデルであるＤＯＭ（Document Object Model）を入力として変換することができる。このＤＯＭの変換は、変換元であるモデルにおける内容の変更を即時に

反映させるもの（以下、このような変換を動的な変換と称す）ではなく、１つのモデルを全体で他のモデルに変換するもの（以下、このような変換を静的な変換と称す）である。すなわち、変換元のモデルにおける変更を変換先のモデルに反映させるには、変更が行われた変換元のモデル全体を一括して変換することによって行う。Ｘａｌａｎ以外にも種々のＸＳＬＴプロセッサが存在するが、そのほとんどは、Ｘａｌａｎと同様の静的な変換を行うものである。

【００１０】

米国マイクロソフト社より提供されているＸＭＬモジュールであるＭＳＸＭＬに付属されたＸＳＬＴプロセッサは、入力である変換元のモデルは静的であって更新がないが、変換ルールを変更することができる。これによって、変換先のモデルとビューを動的に変えることが可能となる。しかしながら、これは実際には静的な変換をルールごとに逐次行っているに過ぎず、変換元の動的な変更には対応していない。

【００１１】

また、TFI Technologyから提供されているＮａｐａは、入力をストリームとして受け取り、それをプログレッシブに変換して、変換結果を出力ストリームに与えるＸＳＬＴプロセッサである。入力ストリームから読まれた変換元モデルにデータが追記されていくという意味では、変換元が動的に変化しているとも言えるが、変換そのものはストリームからストリームへの静的な変換であり、変換元の動的な変更に対応しているわけではない。

【００１２】

さらに、モデル変換器は単体で使われるだけでなく、エディタに組み込まれたプレビュー用モデルの生成にも応用されている。そのようなモデル変換器の機能を持ったエディタとしては、Wattle Softwareから提供されているＸＭＬ　ｗｒｉｔｅｒや、eXcelonから提供されているＥｘｃｅｌｏｎ　Ｓｔｙｌｕｓなどがある。これらのエディタでは、変換元のモデルの編集用に、ソースコードのビュー（ソース表示）によって表示する。ユーザがＸＭＬデータの編集中に、当該編集に基づく更新されたプレビュー画面を参照したい場合は、明示的な操作によってプレビュー画面の更新機能を呼び出す。これに応じて、エディタに搭載された

モデル変換器が、変換元モデルを一括して変換して変換先モデルを新たに作成し、当該変換先モデルのビューであるプレビュー画面を更新する。

【００１３】

【発明が解決しようとする課題】

上述したように、所定のモデルに対し所定のビューを追加しようとする場合であって、かつ当該モデルに対応したインターフェイスを持ったビューが存在しない場合の解決手段は幾つか考えられるが、要求されるインターフェイスを持った必要なビューを作成する方法は、多大な作成コストを要するという問題があった。

【００１４】

また、Adapterパターンなどを用い、インターフェイスの異なるビューをモデルに適応させて使用する方法は、ビュー自体を作成するほどのコストは要しないが、モデル・ビュー間のインターフェイスは、アプリケーションよっては複雑であり、narrowインターフェイス、つまり適応しなければならないオペレーションの最小の集合が大きい場合、Adapterパターンの作成には、やはり多大な作業コストを要することとなっていた。

【００１５】

さらに、モデル変換器を用いて他のモデルに対応したビューを使用する方法では、モデル変換器を作成する作業コストは、インターフェイスの異なるビューをモデルに適応させる方法と比して低い。

しかし、モデル変換器によるモデルの変換は静的な変換であるため、変換元のモデルに変更が加えられた場合に、即時的に変換先のモデルを変更し、当該変更をビューに反映させることができなかった。すなわち、変換元のアプリケーションでは、変換先のアプリケーションにおいて、モデルがどのようなイベントを発生させることによってビューを変更しているのか知ることができないため、変換元のモデルにおける変更部分のみを変換先のビューに反映させる更新を行うことができない。したがって、変換元のモデルを一括して変換先のモデルに変換し、さらに当該新たな変換先のモデルに基づいてビューを作成し直すという過程を経ていた。

【００１６】

ＸＭＬ　ｗｒｉｔｅｒやＥｘｃｅｌｏｎ　Ｓｔｙｌｕｓなどのエディタは、上述したように、アプリケーションのモデルの他にプレビュー用のモデルを作成し、当該プレビュー用のモデルに対応したビューを用いてプレビュー画面を表示している。しかし、アプリケーションで行われた編集作業によりモデルが変更された場合、変換先であるプレビュー用のモデル及びビュー（以下、このモデルとビューとを併せてモデル・ビュー対と称す）に対して即時的に変更が反映されるのではなく、ユーザが明示的に変換先の更新を要求することによって、一括して変換元のモデルを変換先のモデル・ビュー対に変換している。

すなわち、変換先のモデル・ビュー対が変更された場合であっても、変換先におけるモデルからビューにイベントが伝えられて更新された結果、変更後の新しいビューになっているわけではない。したがって、この場合、変更前のモデル・ビュー対は破棄され、更新後の変換元モデルに基づいて生成された、新しい変換先モデル・ビュー対を作り直すといった、効率の悪い手法となっている。

特に変換元モデル、変換先モデルの一方あるいは両方においてデータサイズが大きい場合、プレビューを要求するたびに処理に多大な時間を要することとなる。また、ハードウェアにおいてもメモリ上の大量の記憶空間を消費する。

【００１７】

さらにまた、変換先のモデル・ビュー対において、変換元のモデルにおける変更の内容のみを反映させるのではなく、モデル・ビュー対を新たに作り直すため、変換先のモデル・ビュー対において、編集過程で生じたカーソルの位置や選択領域の位置などのデータを保持している場合、モデル・ビュー対を作り直すたびにこれらの情報が失われることとなる。

【００１８】

これらの課題に対応する手段として、変換元のモデルの変更を変換先のモデルに反映させ、当該変換先のモデルにおいて当該更新を実現するイベントを生成し、このイベントによって変換先のビューを更新する手段が考えられる。

しかしながら、変換元のモデル・ビュー対と変換先のモデル・ビュー対とでは、通常、イベントの定義自体が異なるため、変換元モデルの更新から変換先ビュ

ーを更新するためのイベントを発生させることができない。

【００１９】

そこで、本発明は、モデル変換を用いて所定のモデルから他のアプリケーションのビューを使用する場合に、変換元のモデルに対する変更を動的に変換先のモデル及びビューに反映させ、ビューを更新できるようにすることを目的とする。

【００２０】

また、本発明は、モデル変換を用いて所定のモデルから他のアプリケーションのビューを使用する場合に、変換元のモデルの変更に対応する部分的な変更を変換先のビューに対して行うことによりビューを更新するシステムを提供することを他の目的とする。

【００２１】

【課題を解決するための手段】

上記の目的を達成する本発明は、コンピュータを用いて、モデルとビューとを別個に有するアプリケーションの編集を行うアプリケーション編集装置において、このアプリケーションにおける第１のモデルを編集する編集手段と、この第１のモデルを第２のモデルに変換するモデル変換手段と、この第２のモデルをこの第２のモデルのビューにてディスプレイ装置に表示するビュー表示手段とを備える。そして、このビュー表示手段は、編集手段による第１のモデルの編集に基づいて第２のモデルが更新された場合に、この第２のモデルの更新に基づくイベントを生成するイベント生成手段を備え、このイベント生成手段にて生成されたイベントに基づいてディスプレイ装置に表示されたビューを変更することを特徴とする。

【００２２】

ここで、このビュー表示手段は、編集手段による第１のモデルの編集に基づいて第２のモデルが更新された場合に、この第２のモデルの更新前後の差分を抽出する差分抽出手段をさらに備える構成とすることができる。この場合、イベント生成手段は、差分抽出手段にて抽出された差分に関する情報をパラメータとしてイベントを生成する。

【００２３】

また、このモデル変換手段は、第１のモデルを要素ごとに第２のモデルの対応要素に変換する構成とすることができる。

さらに詳しくは、このモデル変換手段は、第１のモデルにおける要素の変換結果に対応する要素が第２のモデルに存在しない場合に、この変換結果に対応する要素を第２のモデルに付与する構成とすることができる。

このように構成すれば、第２のモデルを更新した場合、更新前後で変化のない要素については更新前の要素をそのまま用いることができ、第２のモデルとそのビューとの間で保持されている情報を保存することができる。

【００２４】

さらにまた、このモデル変換手段は、第１のモデルにおける編集手段により編集された要素を、第２のモデルにおける対応する要素に変換し、変換された要素を用いて第２のモデルを更新する構成とすることができる。

このような構成とすれば、第１のモデルにおいて変更された要素のみを第２のモデルにおける要素に変換し、第２のモデルに反映させることができるので、モデル変換における作業効率を向上させることができる。

【００２５】

また、本発明は、上記のようなアプリケーション編集装置において、アプリケーションにおける第１のモデルを編集する編集手段と、この第１のモデルを第２のモデルに変換するモデル変換手段と、この第２のモデルをこの第２のモデルのビューにてディスプレイ装置に表示するビュー表示手段と、第１のモデルの更新をこの第１のモデルのビューに反映させるためのイベントを、モデル変換に用いられる変換ルールを用いて、第２のモデルのビューを変更するイベントに変換するイベント変換手段とを備え、ビュー表示手段は、このイベント変換手段にて生成されたイベントに基づいて、ディスプレイ装置に表示されたビューを変更する構成とすることができる。

このような構成とすれば、第２のモデルにおいて更新前後の差分を抽出することなく第２のモデルのビューを変更するイベントを生成することができる。

【００２６】

さらにまた、本発明は、コンピュータを用いて、所定のアプリケーションにお

けるモデルを他のアプリケーションにおけるビューにて表示するデータ処理方法において、所定のアプリケーションにおける第１のモデルが更新された場合に、データ記憶手段から他のアプリケーションにおける第２のモデルを読み出し、第１のモデルにおける更新内容を反映させて第２のモデルを更新するステップと、この第２のモデルの更新に基づくイベントを生成し、このイベントに基づいてディスプレイ装置に表示された他のアプリケーションにおけるビューを変更するステップとを含むことを特徴とする。

【００２７】

詳しくは、他のアプリケーションにおけるビューを変更するステップは、第２のモデルにおける更新前後の差分を抽出するステップと、更新前の第２のモデルに抽出された差分に相当する変更を加えることにより前述のイベントを生成するステップと、このイベントに基づいてビューを変更するステップとを含む。

【００２８】

また、第２のモデルを更新するステップは、第１のモデルを要素ごとに第２のモデルの対応要素に変換するステップを含み、かつ他のアプリケーションにおけるビューを変更するステップは、第２のモデルにおける変換された要素ごとにこの第２のモデルにおける更新前後の差分を抽出するステップを含む構成とすることができる。

【００２９】

さらに、他のアプリケーションにおけるビューを変更するステップは、第１のモデルの更新をそのビューに反映させるためのイベントを、第１のモデルを第２のモデルに変換するための変換ルールを用いて、他のアプリケーション、すなわち第２のモデルのビューを変更するイベントに変換するステップを含む構成とすることができる。

【００３０】

また、本発明は、次のように構成されたプログラムとして実現することができる。すなわち、コンピュータを制御して、モデルとビューとを別個に有するアプリケーションを実行するプログラムであって、このプログラムは、このアプリケーションを格納したデータ記憶手段からこのアプリケーションにおけるモデルを

読み出し、このモデルのビューをディスプレイ装置に表示する処理と、このモデルが更新された場合に、このモデルの更新前後の差分を抽出する処理と、抽出された差分に基づいてビューを変更するためのイベントを生成する処理と、生成されたイベントに基づいてディスプレイ装置に表示されているビューを変更する処理とをこのコンピュータに実行させる。

【００３１】

あるいは、コンピュータを制御して、モデルとビューとを別個に有するアプリケーションを編集するプログラムであって、このプログラムは、このアプリケーションにおける第１のモデルに対する編集を行う編集手段と、編集手段にて編集された第１のモデルを第２のモデルに変換するモデル変換手段と、このモデル変換手段により第１のモデルが第２のモデルに変換された場合に、この第２のモデルと直前に変換された第２のモデルとの差分を抽出する差分抽出手段と、この差分抽出手段にて抽出された差分に基づくイベントを生成するイベント生成手段と、第２のモデルをそのビューにてディスプレイ装置に表示すると共に、このイベント生成手段にて生成されたイベントに基づいてディスプレイ装置に表示されたビューを変更するビュー表示手段として、このコンピュータを動作させる。

【００３２】

【発明の実施の形態】

以下、添付図面に示す本実施の形態について本発明を詳細に説明する。

本発明では、モデルとビューとを別個に有するアプリケーションにおいて、モデルの表示を行うために、モデル変換によって他のアプリケーションのビューを使用する手法を対象とする。そして、変換元のモデルに変更があった場合に、モデル変換によって、当該変更を変換先のモデルに反映させる。この後、当該変換先のモデルにおける変更内容に基づいてビューを更新させるためのイベントを生成する。このイベントによって、ビューは、変換元のモデルにおいて変更のない部分についてはそれまでの表示内容を保持しつつ、変換元のモデルの変更に対応する部分の変更を行うといった更新を実現することができる。

【００３３】

〔実施の形態１〕

本実施の形態は、変換元のモデルに変更があった場合に、まず変換元のモデルを変換先のモデルに一括して変換し、次に変換先のモデルにおいて変更前後の差分を抽出して変更内容を特定する。そして、当該変更内容に応じたイベントを生成してビューに送り、当該変更内容をビューに反映させることにより、アプリケーションの編集におけるビューの動的な変更を実現する。

【００３４】

図１は、本実施の形態によるアプリケーション編集装置の実現に好適なコンピュータ装置の構成例を模式的に示した図である。

図１に示すコンピュータ装置は、ＣＰＵ（中央処理装置）１０１と、システムバスを介してＣＰＵ１０１に接続されたＭ／Ｂ（マザーボード）チップセット１０２及びメインメモリ１０３と、ＰＣＩバスなどの高速なバスを介してＭ／Ｂチップセット１０２に接続されたビデオカード１０４、ハードディスク１０５及びネットワークインターフェイス１０６と、さらにブリッジ回路１１０及びＩＳＡバスなどの低速なバスを介してＭ／Ｂチップセット１０２に接続されたフロッピーディスクドライブ１０７、キーボード１０８及びＩ／Ｏポート１０９とを備える。

【００３５】

なお、図１は、本実施の形態によるアプリケーション編集装置としてのコンピュータ装置の構成を例示するに過ぎず、本実施の形態を適用可能であれば、他の種々のシステム構成を取ることが可能である。例えば、ビデオカード１０４の代わりにビデオメモリのみを設け、ＣＰＵ１０１自身が描画命令を実行するようなシステムとしても良い。また、図示はしないが、一般的なコンピュータ装置には、入力手段としてのマウス、音声入出力機構、ＣＤ－ＲＯＭドライブなどが接続される。

【００３６】

図２は、プログラム制御されたＣＰＵ１０１において、本実施の形態によるアプリケーションの編集を行うための機能ブロックを示す図である。

図２を参照すると、本実施の形態は、アプリケーションＡを実行するアプリケーションＡ実行部１０と、アプリケーションＡにおけるモデルＡをアプリケーシ

ョンＢにおけるモデルＢに変換するモデル変換器２０と、アプリケーションＢを実行するアプリケーションＢ実行部３０とを備える。

なお、図２に示すアプリケーションＡ実行部１０、モデル変換器２０及びアプリケーションＢ実行部３０は、図１のメインメモリ１０３にロードされたコンピュータプログラムにより制御されたＣＰＵ１０１にて実現される仮想的なソフトウェアブロックである。当該コンピュータプログラムは、ＣＤ－ＲＯＭやフロッピーディスクなどの記憶媒体に格納して配布したり、ネットワークを介して伝送したりすることにより提供することができる。このようにして提供されたプログラムが、フロッピーディスクドライブ１０７や図示しないＣＤ－ＲＯＭドライブ、ネットワークインターフェイス１０６などを介してメインメモリ１０３に読み込まれ、ＣＰＵ１０１を制御することにより、図１に示したコンピュータ装置が図２に示す各機能を実現する。

【００３７】

ここで、アプリケーションＡは変換元となるアプリケーションを示し、アプリケーションＢは変換先となるアプリケーションを示すに過ぎない。すなわち、個々のアプリケーションが、変換元として扱われる場合はアプリケーションＡとなり、変換先として扱われる場合はアプリケーションＢとなる。

図３は、アプリケーションＡとアプリケーションＢとの関係を説明する図である。

図３に示すように、アプリケーションＡは、ツリー構造のモデルＡとＷＹＳＩＷＹＧ表示のビューＡとを持つものとする。また、アプリケーションＢは、ツリー構造のモデルＢとツリー表示のビューＢとを持つものとする。本実施の形態では、アプリケーションＡにおけるモデルＡを、アプリケーションＢにおけるビューＢを用いて表示するためにモデル変換を行う。なお、アプリケーションＡ、Ｂにおけるビューは１種類である必要はないが、ここでは簡単のため、それぞれ１種類のビューＡ、Ｂを持っているものとする。

【００３８】

図２の構成において、アプリケーションＡ実行部１０は、図１のメインメモリ１０３に読み込まれたアプリケーションＡを実行し、ビデオカード１０４を介し

てディスプレイ装置にモデルＡに基づくビューＡを表示する。また、アプリケーションＡ実行部１０は、モデルＡの編集手段として、キーボード１０８等を介してユーザがビューＡに対して行った編集操作を受け付けて、モデルＡに反映させる。さらに、この編集操作によってモデルＡに対して変更があった場合は、変更があったことをモデル変換器２０に知らせる。

【００３９】

モデル変換器２０は、アプリケーションＡのモデルＡをアプリケーションＢのモデルＢに変換する。モデル変換の方法としては、従来のモデル変換器による方法と同様の方法を用いることができる。アプリケーションＡ実行部１０において、モデルＡの内容が変更された場合は、変更の通知を受けて、当該変更後のモデルＡを入力し、当該変更内容を反映させたモデルＢに変換する。

【００４０】

アプリケーションＢ実行部３０は、図１のメインメモリ１０３に読み込まれたアプリケーションＢを実行し、ビデオカード１０４を介してディスプレイ装置にモデルＢに基づくビューＢを表示する。

また、アプリケーションＢ実行部３０は、モデルＢが更新された場合に、その更新内容に基づいてビューＢを更新する。ここで、ビューＢの更新は、モデルＢの更新前後の差分に基づいて行う。これを実現するため、アプリケーションＢ実行部３０は、差分抽出部３１とイベント生成部３２とを備える。

【００４１】

上記のように、アプリケーションＡ実行部１０においてモデルＡが変更されると、モデル変換器２０により、当該モデルＡの変更内容を反映させてモデルＢが更新される。これに応じて、アプリケーションＢ実行部３０の差分抽出部３１は、当該更新後のモデルＢと更新前のモデルＢとを比較して、差分を抽出する。

イベント生成部３２は、差分抽出部３１によりモデルＢの更新前後の差分が抽出されると、当該差分をパラメータとしてイベントを作成する。すなわち、更新前のモデルＢに対して当該差分を満たす変更を行い、この変更をビューＢに反映させるためのイベントを作成する。

【００４２】

アプリケーションＢ実行部３０は、イベント生成部３２にて作成されたイベントをビューＢに送り、ビューＢを更新する。これによって、ビューＢは、変更のない部分については更新前の表示内容をそのまま表示し、変更部分についてはノードＡの変更内容を反映させた表示内容となるように更新される。

また、アプリケーションＢ実行部３０は、当該変更がなされたモデルＢを保持することとなるが、この場合、イベント生成部３２においてイベントを作成するために行われた変更結果を保持しても良いし、モデル変換器２０による変換において生成されたモデルＢを保持しても良い。イベント生成部３２にて変更されたモデルＢを保持する場合、モデル変換器２０にて生成されたモデルＢを破棄する無駄が生じるが、更新前のモデルＢ及びビューＢの対において、編集過程で生じたカーソルの位置や選択領域の位置などのデータを保持している場合、この情報を保存することができる。

【００４３】

図４は、上述したアプリケーションＡ実行部１０、モデル変換器２０、アプリケーションＢ実行部３０によるモデル及びビューの更新の動作を説明するフローチャートである。また、図５は、かかる更新動作を図３のモデル・ビュー対に対して適用した様子を示す図である。図５中に示された符号は、図４のステップに対応する。

図４に示すように、変換元のビューＡに書き込みが発生すると（ステップ４０１）、アプリケーションＡ実行部１０により、変更されたビューＡに対応してモデルＡが変更される（ステップ４０２）。そして、アプリケーションＡ実行部１０からモデル変換器２０へ、モデルＡの更新が通知される。

次に、アプリケーションＡ実行部１０からの通知を受け取ったモデル変換器２０が、モデルＡにおける変更を反映させたモデルＢを生成し、アプリケーションＢ実行部３０に渡す（ステップ４０３）。

【００４４】

アプリケーションＢ実行部３０は、モデル変換器２０からモデルＡの変更を反映させたモデルＢを受け取り、差分抽出部３１により、モデルＢの更新前後の差分を抽出する（ステップ４０４）。そして、イベント生成部３２により、この差

分をパラメータとするイベントを生成する（ステップ４０５）。このイベントによって、ビューＢが、モデルＡの変更内容を反映させた表示に更新される（ステップ４０６）。

【００４５】

上述した更新前後のモデルＢの差分を用いてビューＢを動的に変更する手法を用いれば、従来のように更新されたモデルＢに基づいてビューＢを作り直す静的な更新に比べて、更新における処理速度が向上する。これにより、アプリケーションＡ実行部１０におけるビューＡに対する編集操作に対し、即時的にビューＢを更新させることが可能となり、ビューＢを確認しながらの編集作業が可能となる。

【００４６】

ここで、ビューＢの更新における処理速度の向上を、処理コストの観点から検討する。

本実施の形態において、モデルＡの変更が発生してからビューＢが更新されるまでのコストVt1は，以下のように定義される。

$$Vt1 = Vconv1 + Vdiff1 + Vupdate + Const \quad (1)$$

式（１）において、

Vconv：モデルの変換コスト。変換元モデル（モデルＡ）のエレメント数をｍとすると計算量とメモリ空間使用量はＯ（ｍ）。

Vdiff：差分作成コスト。変換先モデル（モデルＢ）のエレメント数をｎとすると計算量はＯ（ｎ）。

Vupdate：変換先のビュー（ビューＢ）によるイベントの解釈及び再描画に要するコスト。変換先モデル（モデルＢ）のエレメント数をｎとすると計算量はＯ（ｎ）。

Const：その他（ステップ４０１、４０２、４０５など）の処理コスト。モデルのエレメント数に依存しないのでＯ（１）。

なお、Ｏ（ｍ）、Ｏ（ｎ）は、それぞれｍ、ｎの定数倍以内の時間で、Ｏ（１

）は１が定数なので定数時間内で計算できることを意味する。

【００４７】

一方、従来技術で用いられるような、更新後のモデルＢを反映したビューＢを改めて作り直す手法の場合、ビューＢの再描画がなされるまでのコストVt0は、以下のように定義される。

$$Vt0 = Vconv + Vview + Vdraw + Const \qquad (2)$$

式（２）において、

Vview：変換先ビューＢの作成コスト。メモリ空間使用量、計算量ともにＯ（ｎ）。

Vdraw：変換先ビューＢの初期描画コスト。計算量はＯ（ｎ）。

【００４８】

上の定義式（１）（２）から、Vt0とVt1の比較は、（Vdiff1 + Vupdate）と（Vview + Vdraw）との比較になる。ここで、

$$Vupdate \ll Vvinit$$

という関係が成り立つ。なぜなら、既に初期描画がなされておりイベントを解釈して当該イベントに関わりのある領域を再描画する方が、ビューＢの全ての領域を描画し直すよりも圧倒的に速いからである。よって、Vt0とVt1との比較は、Vdiff1とVviewとの比較に帰着できるが、計算量のオーダーは同じであり正確には優劣が判断できない。しかし、少なくともメモリ空間使用量の点では、Vdiff1の方が優れていることがわかる。

【００４９】

〔実施の形態２〕

上述した実施の形態１では、モデル変換を行う際に、変換先モデル（モデルＢ）全体を作成している。しかしながら、ビューＢの更新に必要な情報はモデルＢにおける更新前後の差分であるため、その他の部分の変換は冗長な処理となって

いる。モデルがツリーなどの構造を持ったオブジェクトの集まりである場合、変更のない個所は変更前のオブジェクトをそのまま再利用したほうが、モデル変換における効率が良い。

そこで、本実施の形態では、モデル変換においても、変換元モデル（モデルＡ）の変更部分に限定して処理を行うこととする。

【００５０】

本実施の形態は、実施の形態１と同様に、図１に示すようなコンピュータ装置にて実現される。

図６は、本実施の形態によるアプリケーションの編集を行うための機能ブロックを示す図である。

図６を参照すると、本実施の形態は、アプリケーションＡを実行するアプリケーションＡ実行部１０と、アプリケーションＡにおけるモデルＡをアプリケーションＢにおけるモデルＢに変換するモデル変換器４０と、アプリケーションＢを実行するアプリケーションＢ実行部３０とを備える。

これらの構成要素のうち、アプリケーションＡ実行部１０及びアプリケーションＢ実行部３０は、図２に示したアプリケーションＡ実行部１０及びアプリケーションＢ実行部３０と同様であるため、同一の符号を付して説明を省略する。また、アプリケーションＡとアプリケーションＢとの関係は、実施の形態１において図３を参照して説明したものと同様である。

なお、図６に示すモデル変換器４０は、図２に示した各構成要素と同様に、図１のメインメモリ１０３にロードされたコンピュータプログラムにより制御されたＣＰＵ１０１にて実現される仮想的なソフトウェアブロックである。

【００５１】

図６の構成において、モデル変換器４０は、モデルＡと変更前のモデルＢとを入力し、変更後のモデルＢを出力する。ただし、最初に変換を行う場合は変更前のモデルＢは存在していないので、モデルＡのみを入力して通常の変換処理を行う。

図７は、モデル変換器４０による変換処理の様子を示す図である。

図７に示すように、モデル変換器４０は、入力したモデルＡと変更前のモデル

Ｂとを要素ごとに比較し、モデルＡの変更に対応する変更をモデルＢに加える。

【００５２】

図７に示した変換処理は、モデルＡの要素をモデルＢの要素に置き換える操作に分解できる。モデル変換器４０は、図６に示すように、この要素の変換を行うための副変換器４１を内蔵する。

図８は、副変換器４１による変換処理の様子を示す図である。

図８に示すように、副変換器４１は、モデルＡにおける変換対象の要素と、変更前のモデルＢにおける対応部分の要素とを入力し、必要な変更をモデルＢの対応部分に加えて出力する。

【００５３】

図９は、モデル変換器４０が副変換器４１による変換処理を統合してモデル変換を行う処理を説明するフローチャートである。

図９に示すように、モデル変換器４０は、モデルＡとモデルＢとを入力した後、モデルＡの要素を１つずつ選択して順次副変換器４１の入力１とする（ステップ９０１）。

副変換器４１は、モデルＡの要素を入力すると、モデルＢのマッピングをたどり、対応する要素を入力２として、さらに入力する（ステップ９０２）。対応する要素がモデルＢに存在しない場合は、「Ｎｕｌｌ」を入力２する。

【００５４】

次に、副変換器４１は、所定の変換ルールに基づいて入力１であるモデルＡの要素をモデルＢの対応する要素に変換し、バッファに一時的に格納する（ステップ９０３）。そして、副変換器４１は、入力２が「Ｎｕｌｌ」であった場合は、バッファに格納された入力１の変換後の要素を出力とし（ステップ９０４、９０５）、「Ｎｕｌｌ」でなかった場合は、入力２をバッファにコピーして出力とする（ステップ９０４、９０６）。すなわち、モデルＡの対応要素がモデルＢに存在する場合は当該対応要素が出力とされ、対応要素がモデルＢに存在しない場合にステップ９０３による当該要素の変換結果が出力とされる。

【００５５】

この後、副変換器４１は、入力１の要素と出力の要素との間にマッピングを作

成し、当該要素に対する変換処理（要素変換）を終了する（ステップ９０７）。

当該要素変換が終了すると、モデル変換器４０は、未処理の要素があるか調べ、あれば上記の処理を繰り返す（ステップ９０８）。そして、モデルＡの全ての要素に対して上記の処理が行われたならば、処理を終了する。

【００５６】

上記のようなモデル変換器４０を用いれば、図１のＣＰＵ１０１におけるキャッシュメモリやメインメモリ１０３における変換作業用の記憶空間の使用量を削減することができる。

また、ステップ９０２において入力１であるモデルＡの要素の変換結果をバッファに格納する際に、当該変換結果とモデルＢの対応要素とを比較し、これらが異なる場合はその違いをモデルＢの更新前後の差分として抽出することにより、アプリケーションＢ実行部３０が行う差分抽出のプロセスを簡略化することができる。すなわち、アプリケーションＢ実行部３０において変更前後のモデルＢを比較して差分を抽出するのではなく、モデル変換器４０における要素変換の際の要素ごとの比較によって抽出された差分を用いて、ビューＢを更新するためのイベントを発生させることができる。

【００５７】

ところで、モデルＡの要素とモデルＢの要素とは必ずしも１対１で対応するとは限らないが、要素の個数の増減（モデルＡの１つの要素にモデルＢの複数の要素の組合せが対応する場合、及びモデルＡの複数の要素の組合せにモデルＢの１つの要素が対応する場合）に対しても、上述した副変換器４１を用いた再帰的な処理によって対応することができる。

具体的なモデル変換の例を挙げて説明する。

図１０は、この種のモデル変換を説明する図である。

この例では、モデルＡ、Ｂが共にツリー構造のオブジェクトとして表現されている場合を取り上げる。このモデルＡ、Ｂは、ツリーの各ノードに、当該ノードの子ノードを示す「child」という配列型の属性が備わっている。

この場合、モデルＡの要素数に対して対応するモデルＢの要素数が増減するような変換の種類は、大きく３種類に分けられる。すなわち、減少型、増加型、倍

化型の３種類である。

【００５８】

まず、減少型について説明する。

減少型の変換の様子を図１０（Ａ）に示す。図１１は、図１０（Ａ）の変換を、要素ごとの処理に分解して示した図である。

図１０（Ａ）及び図１１を参照すると、まず、モデルＡのノードｓ１が、モデルＢのノードｄ１に変換される。次に、ノードｓ２に対応するノードがモデルＢには存在しないのでＮｕｌｌとなる。そして、モデルＡにおけるノードｓ２の子孫が、モデルＢにおけるノードｄ１の子孫に対応づけられる。これにより、次に、ノードｓ２の子孫であるノードｓ３、ｓ４、ｓ５が、それぞれモデルＢのノードｄ２、ｄ３、ｄ４に変換される。

以上の変換を実現する変換ルールを次に示す。

1. s1 -> d1
2. s1.child[0] -> Null
3. s2 -> Null
4. s2.child[0] -> d1.child[0]
5. s2.child[1] -> d1.child[1]
6. s2.child[2] -> d1.child[2]
7. s3 -> d2
8. s4 -> d3
9. s5 -> d4

【００５９】

次に、増加型について説明する。

増加型の変換の様子を図１０（Ｂ）に示す。図１２は、図１０（Ｂ）の変換を、要素ごとの処理に分解して示した図である。

図１０（Ｂ）及び図１２を参照すると、まず、モデルＡのノードｓ６が、モデルＢのノードｄ５、ｄ６に変換される。ここで、ノードｄ５のchild属性として

ノードｄ６が記述される（これによりノードｄ６がノードｄ５の子ノードであることが示される）。そして、モデルＡにおけるノードｓ６の子孫が、モデルＢにおけるノードｄ６の子孫に対応づけられる。これにより、次に、ノードｓ６の子孫であるノードｓ７、ｓ８、ｓ９が、それぞれモデルＢのノードｄ７、ｄ８、ｄ９に変換される。

以上の変換を実現する変換ルールを次に示す。

１．s6 -> d5, d6, d5.child[0]=d6
２．s6.child[0] -> d6.child[0]
３．s6.child[1] -> d6.child[1]
４．s6.child[2] -> d6.child[2]
５．s7 -> d7
６．s8 -> d8
７．s9 -> d9

【００６０】

次に、倍化型について説明する。

倍化型の変換の様子を図１０（Ｃ）に示す。図１３は、図１０（Ｃ）の変換を、要素ごとの処理に分解して示した図である。

図１０（Ｃ）及び図１３を参照すると、まず、モデルＡのノードｓ１０が、モデルＢのノードｄ１０、ｄ１４に変換される。そして、モデルＡにおけるノードｓ１０の子孫が、モデルＢにおけるノードｄ１０及びノードｄ１４の子孫にそれぞれ対応づけられる。これにより、次に、ノードｓ１０の子孫であるノードｓ１１がモデルＢのノードｄ１１、ｄ１５に、ノードｓ１２がモデルＢのノードｄ１２、ｄ１６に、ノードｓ１３がモデルＢのノードｄ１３、ｄ１７に、それぞれ変換される。

以上の変換を実現する変換ルールを次に示す。

１．s10 -> d10, d14

２．s10.child[0] -> d10.child[0], d14.child[0]

３．s10.child[1] -> d10.child[1], d14.child[1]

４．s10.child[2] -> d10.child[2], d14.child[2]

５．s11 -> d11, d15

６．s12 -> d12, d16

７．s13 -> d13, d17

【００６１】

本実施の形態によれば、実施の形態１においてモデルＡの変更が発生するたびに生成されるモデルＢを格納するために消費されていたＣＰＵ１０１のキャッシュメモリやメインメモリ１０３の記憶領域を消費しなくて済む。したがって、モデルの変換コストVconv（式（１）（２）参照）において、メモリ空間使用量がＯ（１）となり、計算量だけがＯ（ｍ）となり、メモリの使用効率が大幅に向上する。

【００６２】

〔実施の形態２の拡張〕

実施の形態２では、モデル変換をモデルの要素ごとに行うことによって、変換元モデル（モデルＡ）における変更のない部分の不要な変換を削減した。しかしながら、モデルＡにおける変更に基づいて発生するアプリケーションＡにおけるイベントなどの情報に基づいて、変換先モデル（モデルＢ）における変更箇所が予測できるならば、モデル変換による新たなモデルＢの生成及びモデルＢにおける更新前後の差分抽出における処理効率を向上させることができる。

【００６３】

すなわち、変換元であるアプリケーションＡ及び変換先であるアプリケーションＢの種類（すなわちモデルＡ及びモデルＢの種類）、及びモデルＡの変更の内容によっては、モデルＢにおける変更箇所が予測できる場合がある。

具体的には、モデルＡの要素に変更が加わったときに、モデル変換の結果において、モデルＢにおける対応要素以外に変更が加わらないことが条件となる。

【００６４】

このような予測ができる場合、モデル変換器４０は、モデルＡ、Ｂにおける全ての要素を副変換器４１にて処理する必要はなく、変更箇所の要素のみを副変換器４１にて処理すれば良いこととなる。

また、アプリケーションＢ実行部３０は、モデルＢ全体における更新前後の差分を抽出する必要はなく、副変換器４１によって処理された要素においてのみ更新前後の差分を抽出すれば良いこととなる。

【００６５】

また、モデルＡの要素数に対して対応するモデルＢの要素数が増減するような変換に対しても、本手法を適用して差分抽出の効率化を図ることができる。

本手法を適用する第１の方法は、モデルＡ、Ｂ間で対応要素が存在しない場合、対応要素が存在する部分まで差分生成範囲を拡大し、当該拡大された差分生成範囲に対して変換処理を行う。ここで、差分生成範囲の拡大は、例えばツリー構造のモデルＡ、Ｂの場合、対応要素が存在しないことが検出された要素からルート方向へ拡大する。

モデルＡの要素数に対して対応するモデルＢの要素数が増減するような変換に対して本手法を適用する第２の方法は、変換ルールを参照して差分生成範囲を決定する方法である。

【００６６】

図１０（Ａ）の減少型の変換において、モデルＡにおけるノードｓ２のchild属性に変更が加わった場合を例に考える。ここで、当該ノードｓ２自体の変換先の要素は存在しないが、第１の方法では、ノードｓ１以下の部分木を差分生成範囲とすることができる。また、第２の方法では、ノードｓ２のchild属性はノードｄ１のchild属性に変換されるというルールから、モデル生成、差分生成の対象ノードをノードｄ１と決定することができる。

【００６７】

図１４は、この条件を満足しない、すなわち、モデルＡにおける変更からモデルＢにおける変更を予測できない形のモデル変換の例を示す図である。

図１４を参照すると、まず、モデルＡのノードｓ１４が、モデルＢのノードｄ１８に変換される。そして、モデルＡにおけるノードｓ１４の子孫が、モデルＢ

におけるノードｄ１８の子孫に対応づけられる。これにより、次に、ノードｓ１４の子孫であるノードｓ１５、ｓ１６、ｓ１７が、それぞれモデルＢのノードｄ１９、ｄ２０、ｄ２１に変換される。

ここで、モデルＡにおけるノードｓ１５の子孫が、モデルＢの対応要素であるノードｄ１９ではなく、ノードｄ２０の子孫に対応づけられる。これによって、ノードｓ１５の子孫であるノードｓ１８が、モデルＢのノードｄ２２（ノードｄ２０の子孫）に変換される。この部分の変換が上述した条件を満足しないため、図１４のモデル変換では、モデルＡの変更からモデルＢの変更箇所を予測できないこととなる。

以上の変換を実現する変換ルールを次に示す。

1．s14 → d18
2．s14.child[0] → d18.child[0]
3．s14.child[1] → d18.child[1]
4．s14.child[2] → d18.child[2]
5．s15 → d19
6．s16 → d20
7．s17 → d21
8．s15.child[0] -> d20.child[0]
9．s18 → d22

上記の変換ルールのうちで、８番目のルールが上記の条件を満足しないため、図１４のモデル変換ではモデルＡの変更からモデルＢの変更箇所を予測することができない。

【００６８】

しかしながら、図１４のようなモデル変換が存在しなければ、モデルＡの変更からモデルＢの変更箇所を予測することが可能であるため、適用範囲を限定して本手法を適用することにより、モデル変換及びモデルＢにおける変更前後の差分抽出において作業コストを大幅に低減することができる。

本手法による作業コストの低下を具体的に考察する。

まず、説明の簡単化のためにエレメント数が増加しない場合を考える。この場合、モデルの変換コストVconv及び差分作成コストVdiffの計算量は、エレメント１つのみを変換、比較すれば良いのでＯ（１）となる。したがって、イベント解釈及び再描画に要するコストVupdateのみが定数でないＯ（ｎ）となる。したがって、上記の式（２）で定義された、更新後のモデルＢを反映したビューＢを改めて作り直す手法における作業コストVt0に比べて、大幅な作業コストの低減が見込まれる。

次に、エレメント数が増加した場合は、上述したように差分生成範囲を拡大しなければならないが、この範囲は変換ルールなどによって決まっている。したがって、モデルの変換コストVconv及び差分作成コストVdiffの計算量は、共にＯ（１）となる。したがって、上記の式（２）で定義された作業コストVt0に比べて、大幅な作業コストの低減が見込まれる。

【００６９】

〔実施の形態３〕

次に、変換先モデル（モデルＢ）の更新前後の差分を抽出することなくモデルＡの変更に基づいてビューＢを更新するイベントを発生させる手法について説明する。

アプリケーションＡにおけるモデルＡを表示するためにアプリケーションＢにおけるビューＢを使用するためにモデル変換を行う状況で、アプリケーションＡにおける編集作業をビューＢに反映させることは、変換元のモデルＡが変更された場合に、変換先のビューＢに対してどのようなイベントを発生させれば良いかという問題に帰着できる。

上述した実施の形態１、２では、モデルＡの変更に伴ってモデル変換によりモデルＢを更新し、当該モデルＢにおいて更新前後の差分を抽出し、この差分に基づいてモデルＢにおける変更内容を特定し、ビューＢを更新するためのイベントを発生させていた。

これに対し、アプリケーションＡにおいてビューＡを更新するために生成されるイベントを、アプリケーションＢにおいてビューＢを更新するためのイベント

に変換することにより、直接的にビューＢを更新するためのイベントを生成することもできる。

【００７０】

本実施の形態は、実施の形態１と同様に、図１に示すようなコンピュータ装置にて実現される。

図１５は、本実施の形態によるアプリケーションの編集を行うための機能ブロックを示す図である。

図１５を参照すると、本実施の形態は、アプリケーションＡを実行するアプリケーションＡ実行部１０と、アプリケーションＡにおけるモデルＡをアプリケーションＢにおけるモデルＢに変換するモデル変換器２０と、アプリケーションＡにおいてビューＡを更新するイベントをアプリケーションＢにおけるビューＢを更新するイベントに変換するイベント変換器５０と、アプリケーションＢを実行するアプリケーションＢ実行部３０とを備える。

これらの構成要素のうち、アプリケーションＡ実行部１０、モデル変換器２０及びアプリケーションＢ実行部３０は、図２に示したアプリケーションＡ実行部１０、モデル変換器２０及びアプリケーションＢ実行部３０と同様であるため、同一の符号を付して説明を省略する。また、アプリケーションＡとアプリケーションＢとの関係は、実施の形態１において図３を参照して説明したものと同様である。

なお、図１５に示すイベント変換器５０は、図２に示した各構成要素と同様に、図１のメインメモリ１０３にロードされたコンピュータプログラムにより制御されたＣＰＵ１０１にて実現される仮想的なソフトウェアブロックである。

【００７１】

図１５の構成において、イベント変換器５０は、アプリケーションＡでモデルＡが更新された場合に、これに伴ってビューＡを更新するために生成されるイベントを入力する。そして、当該変換元のイベントとモデル変換器２０がモデルＡをモデルＢに変換するために用いる変換ルールとに基づき、アプリケーションＢのビューＢにおいてモデルＡの変更を反映させた更新を行うためのイベントを生成する。

なお、本実施の形態では、イベント変換器５０によってビューＢを更新するためのイベントが生成されるため、アプリケーションＢ実行部３０は、当該イベントに基づいてビューＢを更新すれば良い。したがって、図２に示した差分抽出部３１及びイベント生成部３２は不要となる。

【００７２】

図１６は、上述したアプリケーションＡ実行部１０、モデル変換器２０、イベント変換器５０、アプリケーションＢ実行部３０によるモデル及びビューの更新の動作を説明するフローチャートである。また、図１７は、かかる更新動作を図３のモデル・ビュー対に対して適用した様子を示す図である。図１７中に示された符号は、図１６のステップに対応する。

図１６に示すように、変換元のビューＡに書き込みが発生すると（ステップ１６０１）、アプリケーションＡ実行部１０により、変更されたビューＡに対応してモデルＡが変更される（ステップ１６０２）。そして、アプリケーションＡ実行部１０からモデル変換器２０へ、モデルＡの更新が通知される。

次に、アプリケーションＡ実行部１０からの通知を受け取ったモデル変換器２０が、モデルＡにおける変更を反映させたモデルＢを生成し、アプリケーションＢ実行部３０に渡す（ステップ１６０３）。

【００７３】

また、アプリケーションＡ実行部１０において、モデルＡの変更をビューＡに反映させるためのイベントが生成される（ステップ１６０４）。そして、イベント変換器５０が、アプリケーションＡ実行部１０により生成されたイベントとモデル変換ルールとを用いて、モデルＢの変更をビューＢに反映させるためのイベントを生成し、アプリケーションＢ実行部３０へ通知する（ステップ１６０５）。そして、アプリケーションＢ実行部３０において、このイベントによって、ビューＢが、モデルＡの変更内容を反映させた表示に更新される（ステップ１６０６）。

【００７４】

ところで、モデルＡ、Ｂがツリー構造のモデルである場合、一般的な変換ルールは、１対１でノードが対応する変換（通常）、ノードの削除、ノードの挿入及

びノードに付加された属性の変更（入れ替え）の種類であり、複雑な変更であってもこれらの組合せで表現できる（この他、特殊な場合として、実施の形態２において説明した図１０に示すノード（要素）の個数が増減する変換がある）。

図１８は、これらの変換の様子を示す図である。

これらのうち、ツリー構造が変化する３種類の変換ルール、すなわちノードの削除、ノードの挿入及びノードに付加された属性の変更に関して、モデルＡの該当要素に適用される変換ルールに基づき、モデルＢに対してどのようなイベントを発生させるかについて説明する。

【００７５】

変換元であるモデルＡの該当要素（以下、変換元要素）をdとし、変換先であるモデルＢにおける対応要素（以下、変換先要素）を得るためのマッピングをmapとし、要素dから得られる変換先要素をmap(d)とする。変換元、変換先の各要素は、それぞれ、親を示す「parent」、子供を示す配列型の「children」、自分が親に対して何番目の子供かを示す「index」という属性を持つと仮定する。また、モデルＡで発生するイベント（以下、変換元イベント）をse、モデルＢで発生させるイベント（以下、変換先イベント）をdeとし、それぞれ、変更対象のノードを示す「target」、挿入・削除の対象を示す「element」、挿入・削除の行われた個所を示す「index」、イベントの種類を示す「type（REMOVE、INSERT、CHANGEDのいずれか）」という４つの属性を持っているものとする。

【００７６】

以上のような条件で、イベント変換器５０は、変換元イベントseからマッピングmapを用いて、次のようにして変換先イベントdeの属性を決定する。

・de.type = f(se.type, se.target, se.element)　ただし、fは、変換元要素dにおける変更の種類、変更対象、挿入・削除対象から変換先の変更の種類を得る関数。

・de.target = map(se.target)　ただし、この値がＮｕｌｌの場合は、変換先イベントdeを破棄する（イベントを生成しない）。

・・de.element = map(se.element)

・de.index = map(se.element).index

【００７７】

以下、上述した実施の形態を具体的なアプリケーションに適用した例について説明する。

〔適用例１〕

本適用例では、アプリケーションとして、ＨＴＭＬのＤＯＭを、米国サン・マイクロシステムズ社より提供されているJava 2 Platform, Standard Edition付属のテキストコンポーネントにマッピングする方法で作られたＨＴＭＬエディタを挙げる。

図１９は、このＨＴＭＬエディタの内部構造を示す図である。

図１９において、モデルHTMLDOMは、ＨＴＭＬにおけるDOMのインターフェイスを持ったツリー構造のモデルである。

また、ビューJTree(0)（ツリー表示）は、javax.swing.JTreeのインスタンスである。モデルTreeNodeは、ビューJTree(0)のモデルであり、ＤＯＭから変換されたjavax.swing.tree.TreeNodeインターフェイスを実装している。

また、ビューJEditorPane（ＷＹＳＩＷＹＧ表示）は、javax.swing.JEditorPaneのインスタンスである。モデルSwingDocumentは、ビューJEditorPaneのモデルであり、ＤＯＭから変換されたjavax.swing.text.Documentインターフェイスを実装する。さらに、このモデルSwingDocumentは、TreeNodeインターフェイスも実装しており、これに応じてツリー表示のビューJTree(1)を持っている。

【００７８】

ここで、モデルHTMLDOMとモデルTreeNodeとの間の変換は単純な１対１の変換であり、したがって、ビューJTree(0)は、モデルHTMLDOMのビューに等しい。これに伴って、以下の説明では、モデルHTMLDOMからモデルSwingDocumentへの変換を行う場合について説明する。また、モデル変換の手法は、上述した実施の形態２の手法（すなわち、モデルにおける対応要素ごとに変換を行う手法）とする。

【００７９】

図２０は、モデルHTMLDOMとしてＩＢＭ社のウェブページ（http://www.ibm.com/）を用い、ビューJTree(0)で表示した画像イメージを示す図である。図２１は

、同ウェブページをビューJTree(1)で表示した画像イメージを示す図である。また図２２は、同ウェブページをビューJEditorPaneで表示した画像イメージを示す図である。

【００８０】

ここで、図２２において中央付近に表示された、「Power packed offer」という題を含むイメージデータを示すＤＯＭノード（IMG0）を削除する動作を行うものとする。

図２３は、ノード（IMG0）を削除する前のモデルHTMLDOMにおける当該ノード（IMG0）付近の状態を示す図である。また、図２４は、このモデルHTMLDOMから変換されたモデルSwingDocumentにおける対応要素であるノード（IMG1）付近の状態を示す図である。なお、図２３において、モデルHTMLDOMのイメージは、ツリー表示を行うビューJTree(0)のイメージを用いて表現している。同様に、図２４において、モデルSwingDocumentのイメージは、ツリー表示を行うビューJTree(0)を用いて表現している。

図２３において、ノード（IMG0）の親ノードはノード（Anchor0）であり、図２４において、ノード（IMG1）の親ノードはノード（Anchor1）である。

【００８１】

モデルHTMLDOMからモデルSwingDocumentへ変換する変換ルールは、次のようになっている。

1．TABLE → DomTableElement

2．TABLE.child[0] → Null

3．TBODY → Null

4．TBODY.child[0] → DomTableElement.child[0]，TBODY.child[1] →DomTableElement.child[1],...

5．TR → DomTableRowElement

6．TD → DomTableCellElement

7．A → DomCharacterElement, DomCharacterElement.lastChild = EmptyElement （左項は最後の子供を示す）

８．IMG → DomSpecialElement

【００８２】

図２５は、図２３からノード（IMG0）を削除した状態を示す図である。また、図２６は、このモデルHTMLDOMから変換されたモデルSwingDocumentにおける対応要素であるノード（Anchor1）付近の状態を示す図である。さらに、図２７は、図２６に示すモデルSwingDocumentに対するビューJEditorPaneによる画像イメージを示す図である。

図２５、図２６を参照すると、モデルHTMLDOMにおいてノード（IMG0）が削除されたことに伴い、モデルSwingDocumentにおいてもノード（IMG1）が削除されている。さらに、図２７を参照すると、ビューJEditorPaneにおいて、ノード（IMG1）に対応するイメージデータが削除されていることがわかる。

【００８３】

以上の例では、ノード（IMG0）が削除される変更が発生した変換元モデル（HTMLDOM）の要素がノード（Anchor0）であった。そこで、モデル変換器４０は、副変換器４１を用いて、モデルHTMLDOMのノード（Anchor0）をモデルSwingDocumentのノード（Anchor1）に変換する。

次に、アプリケーションＢ実行部３０の差分抽出部３１が、当該変換前に保持していたノード（Anchor1）とモデル変換器４０から受け取ったノード（Anchor1）とを比較する。この比較によって、ノード（Anchor1）の第１子であるノード（IMG1）が削除されていることがわかり、これが差分として抽出される。

次に、アプリケーションＢ実行部３０のイベント生成部３２が、この差分を反映させるためのイベントを生成し、ビューJEditorPaneに投げる。このイベントは、javax.swing.event.DocumentEventインターフェイスを実装するインスタンスであり、実際の内容は次のようになっている。

DocumentEvent

・イベント発生個所の文書先頭からのオフセット：405

・イベント対象領域の長さ：１

・イベントの種類: REMOVE

・イベント発生エレメント: Anchor1

・変更後に保持している子供のエレメント：0th child: EmptyElement[405, 406, name:content]

・イベントの対象となる子供のインデックス: ０

・取り除かれた子供のエレメント: IMG0

このイベントをビューJEditorPaneが受け取ると、表示内容が図２７に示したように更新される。

【００８４】

〔適用例２〕

本適用例では、ＸＳＬによって変換されたＨＴＭＬのプレビュー機能を持つＸＭＬエディタを挙げる。なお、本適用例においても、モデル変換は実施の形態２の手法によるものとする。

図２８は、このＸＭＬエディタの内部構造を示す図である。

図２８において、モデルTreeNode及びビューTreeEditorからなるモデル・ビュー対は、通常のＸＭＬのツリーエディタ（例えば、ＩＢＭ社のXeena）である。ツリーエディタのモデルTreeNodeは、モデル変換器４０であるＸＳＬＴプロセッサによって、ＨＴＭＬのＤＯＭ（モデルHTMLDOM）に変換される。また、モデルHTMLDOMは、ビューHTML ViewerにてＷＹＳＩＷＹＧ表示される。ビューHTML Viewerには、一般的なウェブブラウザを用いることができる。

【００８５】

ツリーエディタによって、モデルTreeNodeに変更が加えられた場合、ＸＳＬＴプロセッサに変更箇所が通知され、ＸＳＬＴプロセッサによる変換を経て、モデルHTMLDOMにおける対応箇所が変更される。

次に、アプリケーションＢ実行部３０によって、当該変更前後のモデルHTMLDOMの差分が計算される。そして、この差分をパラメータとして、ビューHTML Viewerを更新するためのイベントが生成され、ビューHTML Viewerに投げられる。ビューHTML Viewerは、受け取ったイベントに基づいて、モデルTreeNodeの変更を

反映させるように更新される。

【００８６】

上記のＸＭＬエディタを用いた動作例として、図２９に示す文書型のＸＭＬデータを、携帯情報端末などで用いられるコンパクトＨＴＭＬに変換してプレビューする例を考える。

図３０は、この変換を実現するＸＳＬスタイルシートの例を示す図である。

また、図３１は、変換元モデル（モデルＡ）であるＸＭＬデータの例を示す図、図３２は、図３１のモデルを変換して得られる変換先モデル（モデルＢ）であるコンパクトＨＴＭＬデータを示す図である。ただし、図３１、図３２は、それぞれソース表示のビューＡ、Ｂにて表示された状態が示されている。

【００８７】

ここで、変換元モデルであるＸＭＬデータ（図3.1）に対する変更として、contentエレメントの次に、以下の文書の断片で示された部分木を挿入する。

```
<statement subject="OK">
  <content>OK</content>
</statement>
```

この変更に応じて、モデル変換器４０であるＸＳＬプロセッサが、ＸＭＬデータ（図３１）中のstatementエレメントに関して変換を行い、当該変更を反映させたコンパクトＨＴＭＬデータを生成する。図３３は、生成されたコンパクトＨＴＭＬデータを示す図である。

【００８８】

この変換の際に、１番目のulエレメント以下の差分計算がなされ、２番目のulエレメントにliエレメントが付け加えられたことがわかる。そこで、アプリケーションＢ実行部３０が、次に示すようなＤＯＭの更新イベントを生成し、ビューHTML Viewerを更新する。

```
MutationEvent
```

イベントの種類：INSERT

イベント発生エレメント：２番目のulエレメント

イベントの対象となる子供のインデックス：０

付け加わった子供のエレメント：２番目のliエレメント

【００８９】

〔適用例３〕

以上の適用例では、モデルがツリー構造である場合について説明したが、本実施の形態は、ツリー構造以外の構造を持ったモデルの変換に対しても応用可能である。本適用例では、変換元モデル（モデルＡ）が２次元配列であり、これをＨＴＭＬのテーブルに変換する例を挙げる。

図３４は、モデルＡに変更を加える前のモデルＡ、Ｂを示す図である。

図３４において、モデルＡは表形式によるビューで示し、モデルＢはＨＴＭＬのテーブルをツリー表示によるビューで示している。

【００９０】

図３４に示したモデルＡにおいて、当期利益の列を削除する変更を行う場合を考える。

図３５は、当該変更が行われた場合のモデルＡ、Ｂを示す図である。

図３５のモデルＡに示すように、当期利益の列が削除されている。かかる変更がモデルＡに行われると、モデル変換によって変更内容がモデルＢに反映される。ここで、モデルＢにおけるTABLEエレメント以下の変更前後の差分を作成すると、それぞれのTRエレメントから、最も右に位置していたエレメント（TH/TDエレメント）が削除されたことがわかる。これにより、次に示すイベントが、それぞれのTRエレメントに対して発生する。

MutationEvent

イベントの種類：REMOVE

イベント発生エレメント：TRエレメント

イベントの対象となる子供のインデックス：３

取り除かれた子供のエレメント：TD(最初のTRに対してはTH)エレメント

以上の４つのイベントは、同時に発生するものであるが、一般にはビューとモデルと間でイベントの並列な通知はできないので、直列化して順番にビューに対して通知する。

【００９１】

【発明の効果】

以上説明したように、本発明によれば、モデル変換を用いて所定のモデルから他のアプリケーションのビューを使用する場合に、変換元のモデルに対する変更を即時的に変換先のモデル及びビューに反映させ、ビューを更新することが可能となる。

【００９２】

また、本発明によれば、モデル変換を用いて所定のモデルから他のアプリケーションのビューを使用する場合に、変換元のモデルの変更に対応する部分的な変更を変換先のビューに対して行うことによりビューを更新することが可能となる。

【図面の簡単な説明】

【図１】　実施の形態１によるアプリケーション編集装置の実現に好適なコンピュータ装置の構成例を模式的に示した図である。

【図２】　本実施の形態によるアプリケーションの編集を行うための機能ブロックを示す図である。

【図３】　本実施の形態における変換元のアプリケーションと変換先のアプリケーションとの関係を説明する図である。

【図４】　本実施の形態におけるモデル及びビューの更新の動作を説明するフローチャートである。

【図５】　図４に示す更新動作を図３のモデル・ビュー対に対して適用した様子を示す図である。

【図６】　実施の形態２によるアプリケーションの編集を行うための機能ブロックを示す図である。

【図７】　本実施の形態におけるモデル変換器による変換処理の様子を示す図である。

【図８】　本実施の形態における副変換器による変換処理の様子を示す図である。

【図９】　本実施の形態におけるモデル変換器が副変換器による変換処理を統合してモデル変換を行う処理を説明するフローチャートである。

【図１０】　変換前後で要素の数が増減するモデル変換を説明する図である。

【図１１】　図１０（Ａ）の変換を要素ごとの処理に分解して示した図である。

【図１２】　図１０（Ｂ）の変換を要素ごとの処理に分解して示した図である。

【図１３】　図１０（Ｃ）の変換を要素ごとの処理に分解して示した図である。

【図１４】　モデルＡにおける変更からモデルＢにおける変更を予測できない形のモデル変換の例を示す図である。

【図１５】　実施の形態３によるアプリケーションの編集を行うための機能ブロックを示す図である。

【図１６】　本実施の形態におけるモデル及びビューの更新の動作を説明するフローチャートである。

【図１７】　図１６に示す更新動作を図３のモデル・ビュー対に対して適用した様子を示す図である。

【図１８】　一般的な変換ルールによるモデル変換を説明する図である。

【図１９】　適用例１におけるＨＴＭＬエディタの内部構造を示す図である。

【図２０】　本適用例におけるビューJTree(0)の表示例を示す図である。

【図２１】　本適用例におけるビューJtree(1)の表示例を示す図である。

【図２２】　本適用例におけるビューJEditorPaneの表示例を示す図である。

【図２３】　ノード（IMG0）を削除する前のモデルHTMLDOMにおけるノード（IMG0）付近の状態を示す図である。

【図２４】　図２３のモデルHTMLDOMから変換されたモデルSwingDocumentにおける対応要素であるノード（IMG1）付近の状態を示す図である。

【図２５】　図２３からノード（IMG0）を削除した状態を示す図である。

【図２６】　図２５のモデルHTMLDOMから変換されたモデルSwingDocumentを示す図である。

【図２７】　図２６に示すモデルSwingDocumentに対するビューJEditorPaneによる画像イメージを示す図である。

【図２８】　適用例２におけるＸＭＬエディタの内部構造を示す図である。

【図２９】　ＸＭＬデータの例を示す図である。

【図３０】　モデル変換を行うＸＳＬスタイルシートの例を示す図である。

【図３１】　本適用例における変換元モデル（モデルＡ）であるＸＭＬデータの例を示す図である。

【図３２】　図３１のモデルを変換して得られる変換先モデル（モデルＢ）であるコンパクトＨＴＭＬデータを示す図である。

【図３３】　図３１のモデルに対する変更を反映させたコンパクトＨＴＭＬデータを示す図である。

【図３４】　適用例３において、モデルＡに変更を加える前のモデルＡ、Ｂを示す図である。

【図３５】　図３４のモデルＡに対して変更を加えた後のモデルＡ、Ｂを示す図である。

【図３６】　従来技術における基本的なモデルとビューとの分割を示す図である。

【図３７】　図３６のアプリケーションにおいて、所定のビューにおける変更がモデル及び他のビューに反映する様子を示す図である。

【図３８】　Adapterパターンを模式的に例示する図である。

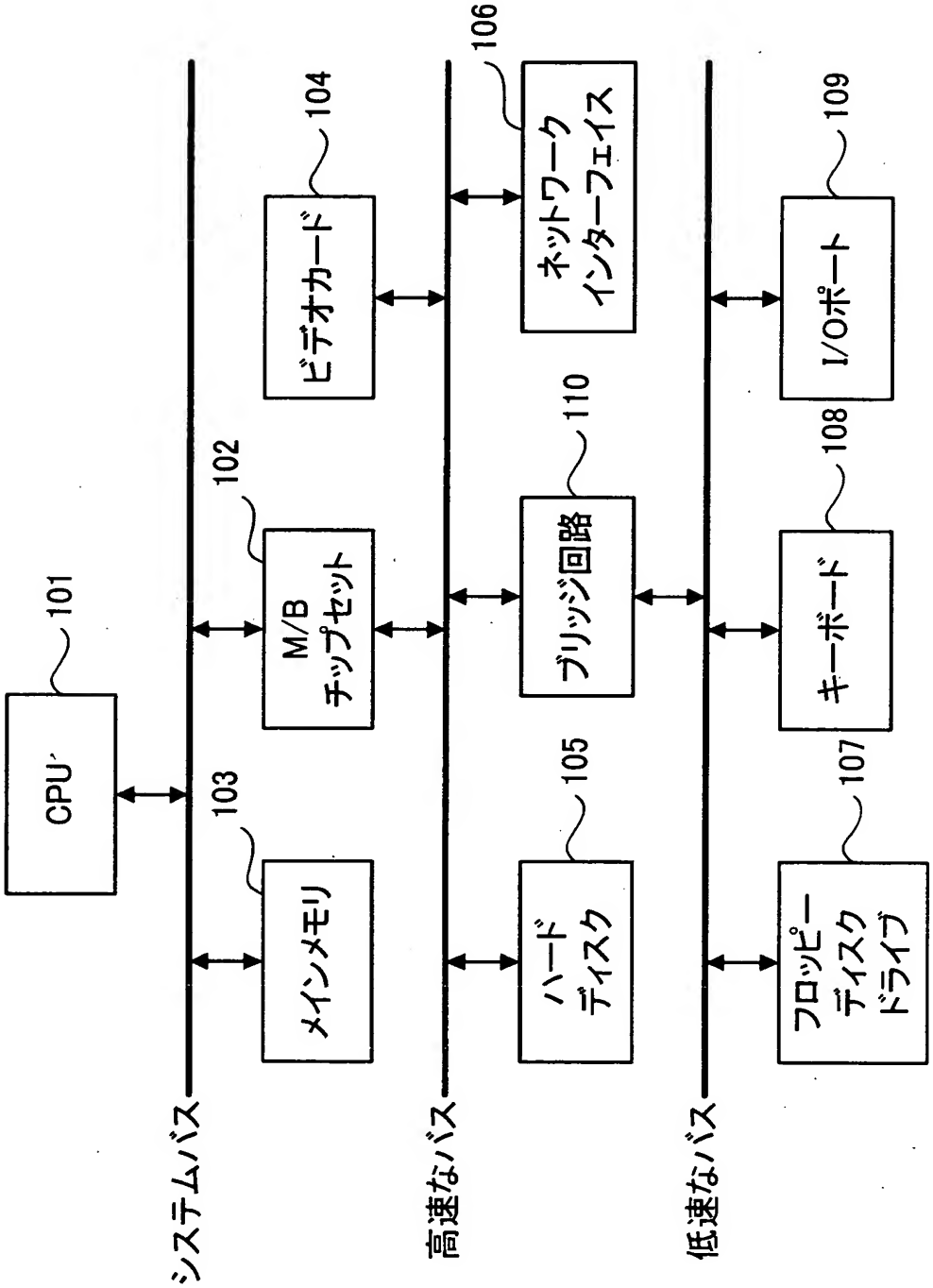
【図３９】　モデル変換により他のアプリケーションのビューを用いて表示を行う方法を説明する図である。

【符号の説明】

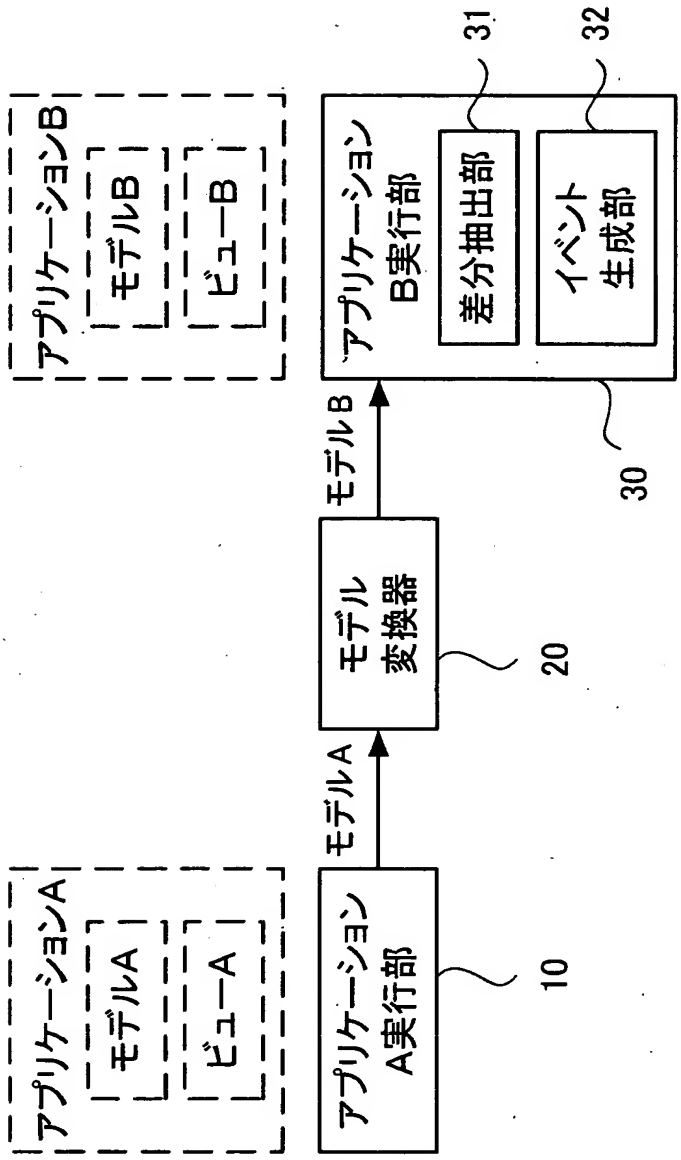
１０…アプリケーションＡ実行部、２０、４０…モデル変換器、３０…アプリケーションＢ実行部、３１…差分抽出部、３２…イベント生成部、４１…副変換器、５０…イベント変換器

【書類名】　　図面

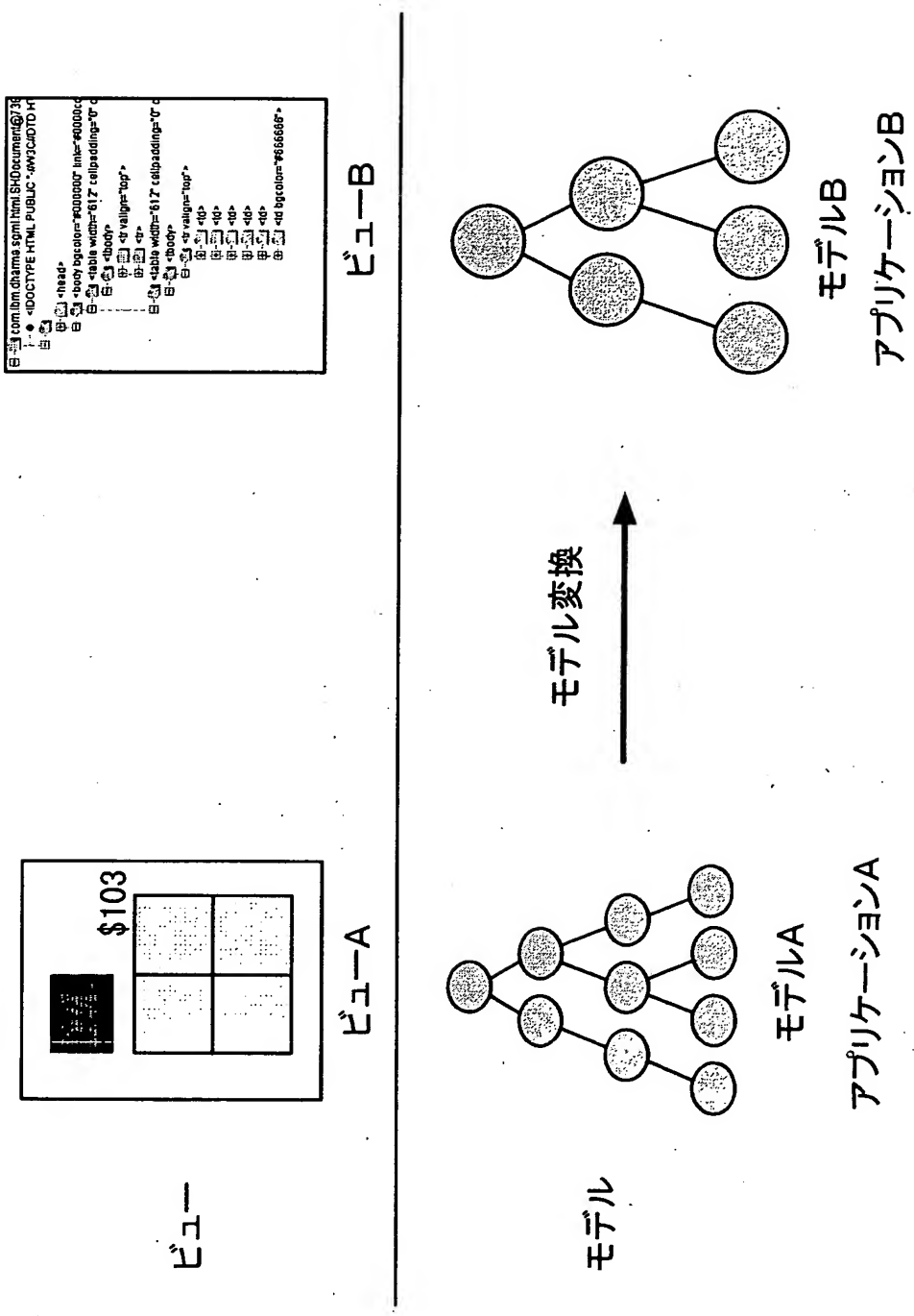
【図１】


CPU
101
M/B
チップセット
102
メインメモリ
103
ビデオカード
104
ハード
ディスク
105
ネットワーク
インターフェイス
106
フロッピー
ディスク
ドライブ
107
ブリッジ回路
110
キーボード
108
I/Oポート
109
システムバス
高速なバス
低速なバス

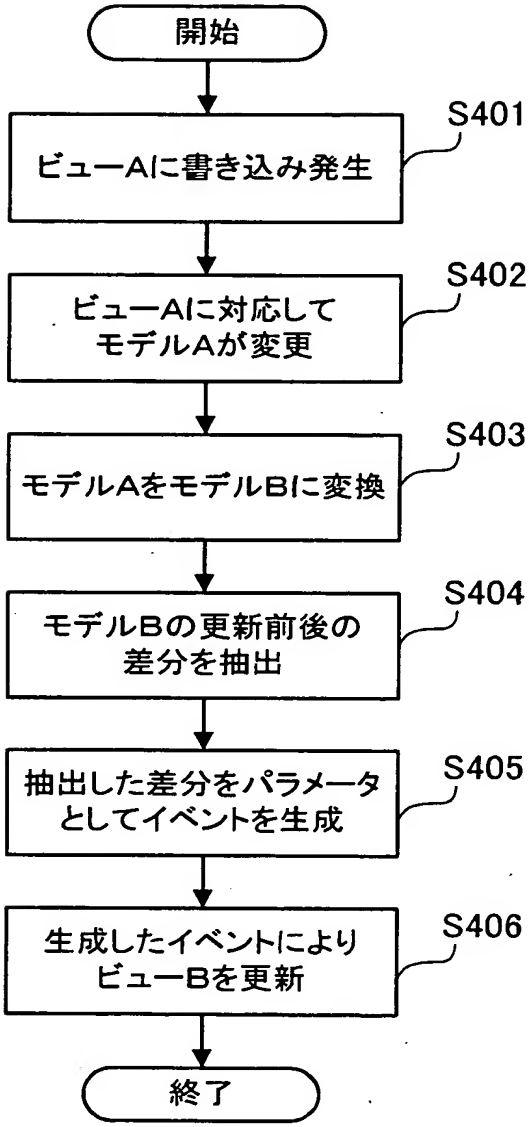
【図２】


アプリケーションA
モデルA
ビューA
アプリケーション
A実行部
10
モデルA
モデル
変換器
20
モデルB
アプリケーションB
モデルB
ビューB
アプリケーション
B実行部
差分抽出部
31
イベント
生成部
32
30

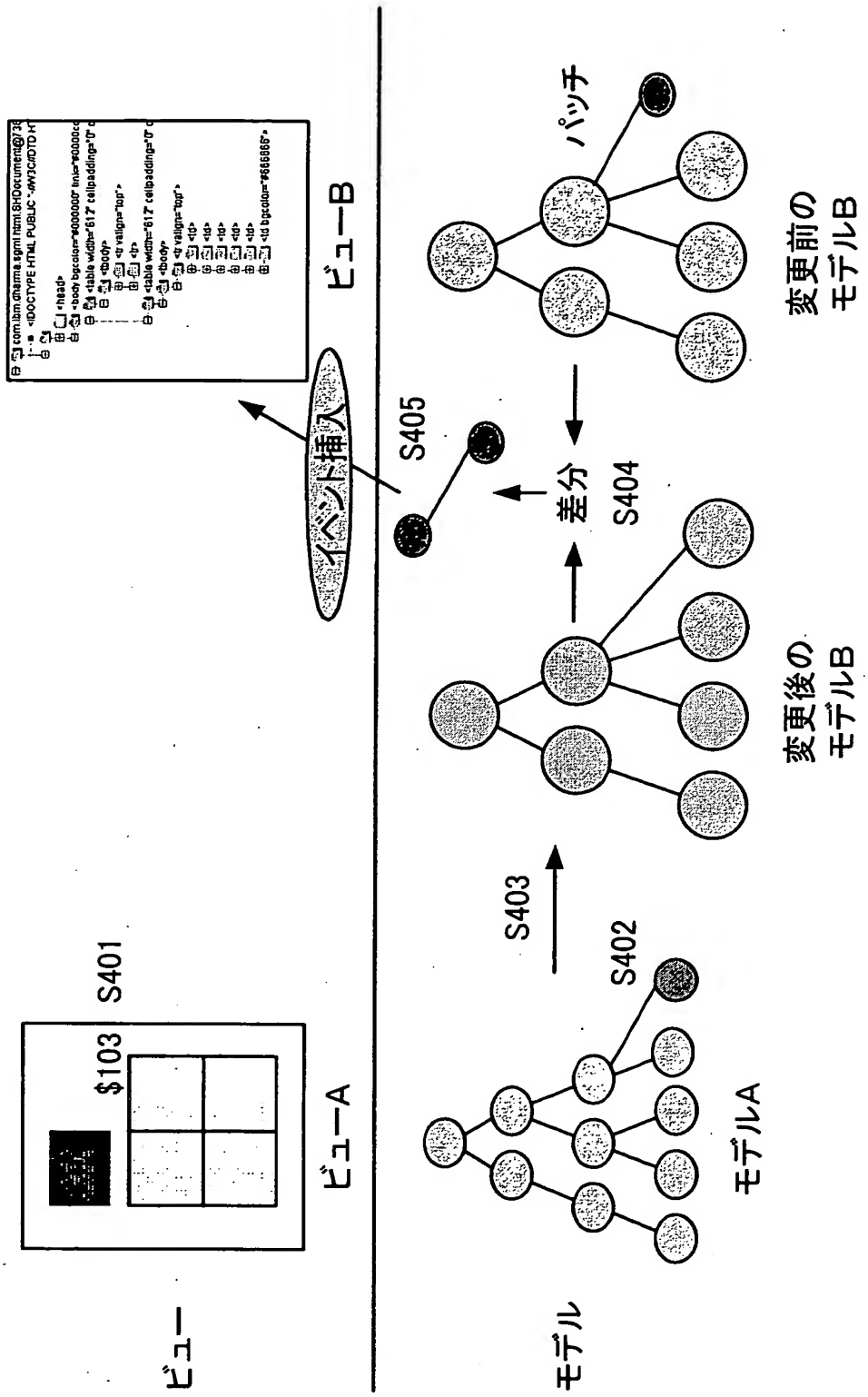
【図３】



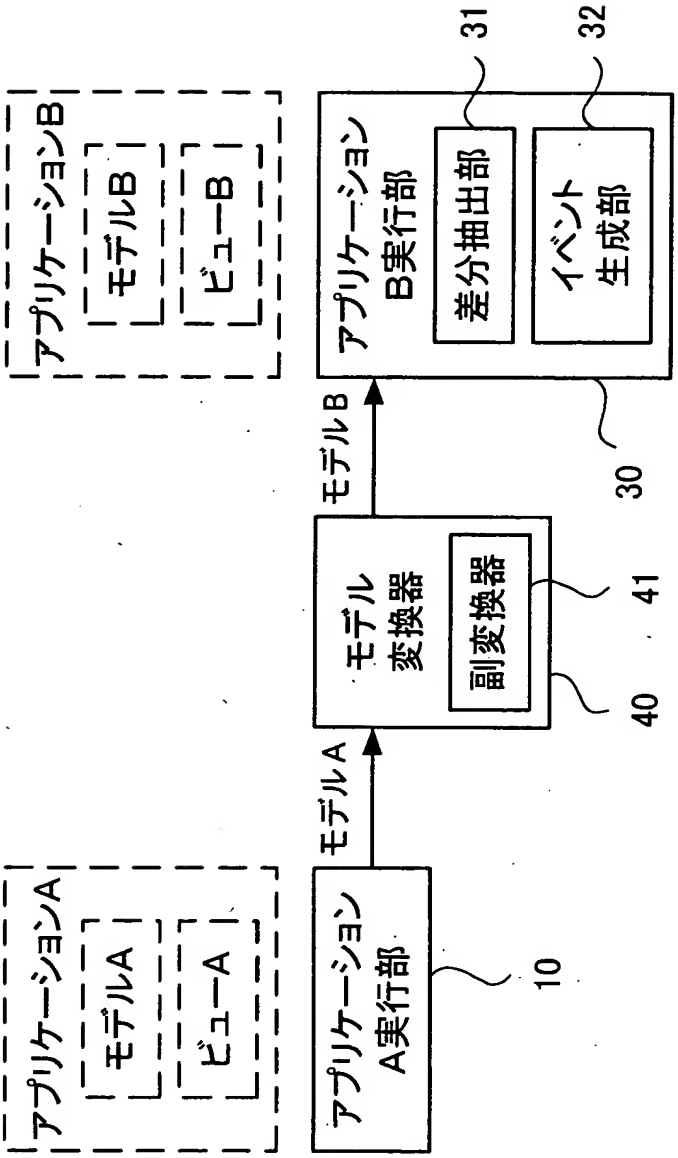
【図４】


開始
S401
ビューAに書き込み発生
S402
ビューAに対応して
モデルAが変更
S403
モデルAをモデルBに変換
S404
モデルBの更新前後の
差分を抽出
S405
抽出した差分をパラメータ
としてイベントを生成
S406
生成したイベントにより
ビューBを更新
終了

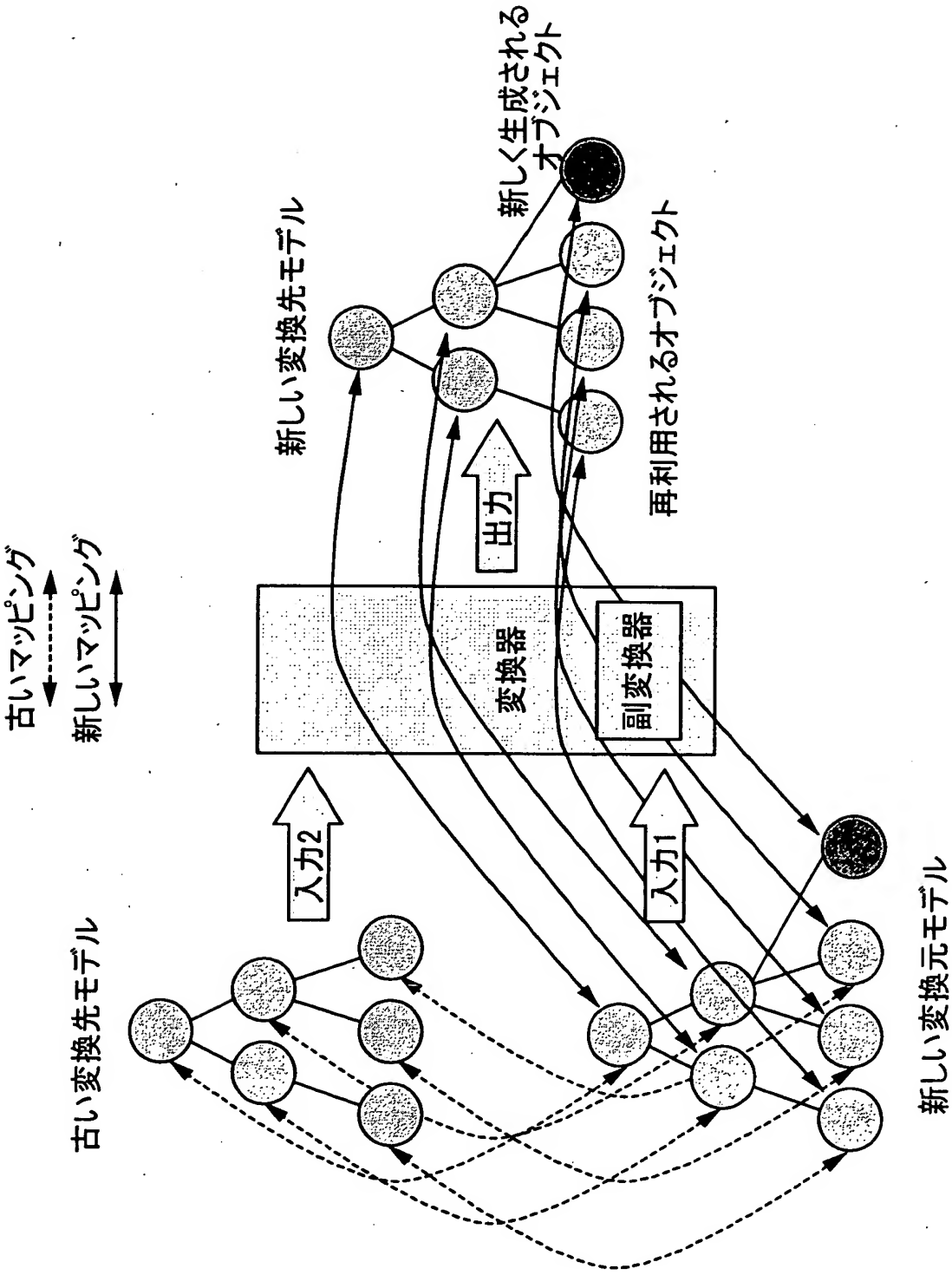
【図５】


ビュー
モデル
ビューA
ビューB
$103
S401
S402
S403
S404
S405
S406
モデルA
変更後の
モデルB
差分
パッチ
変更前の
モデルB
イベント挿入

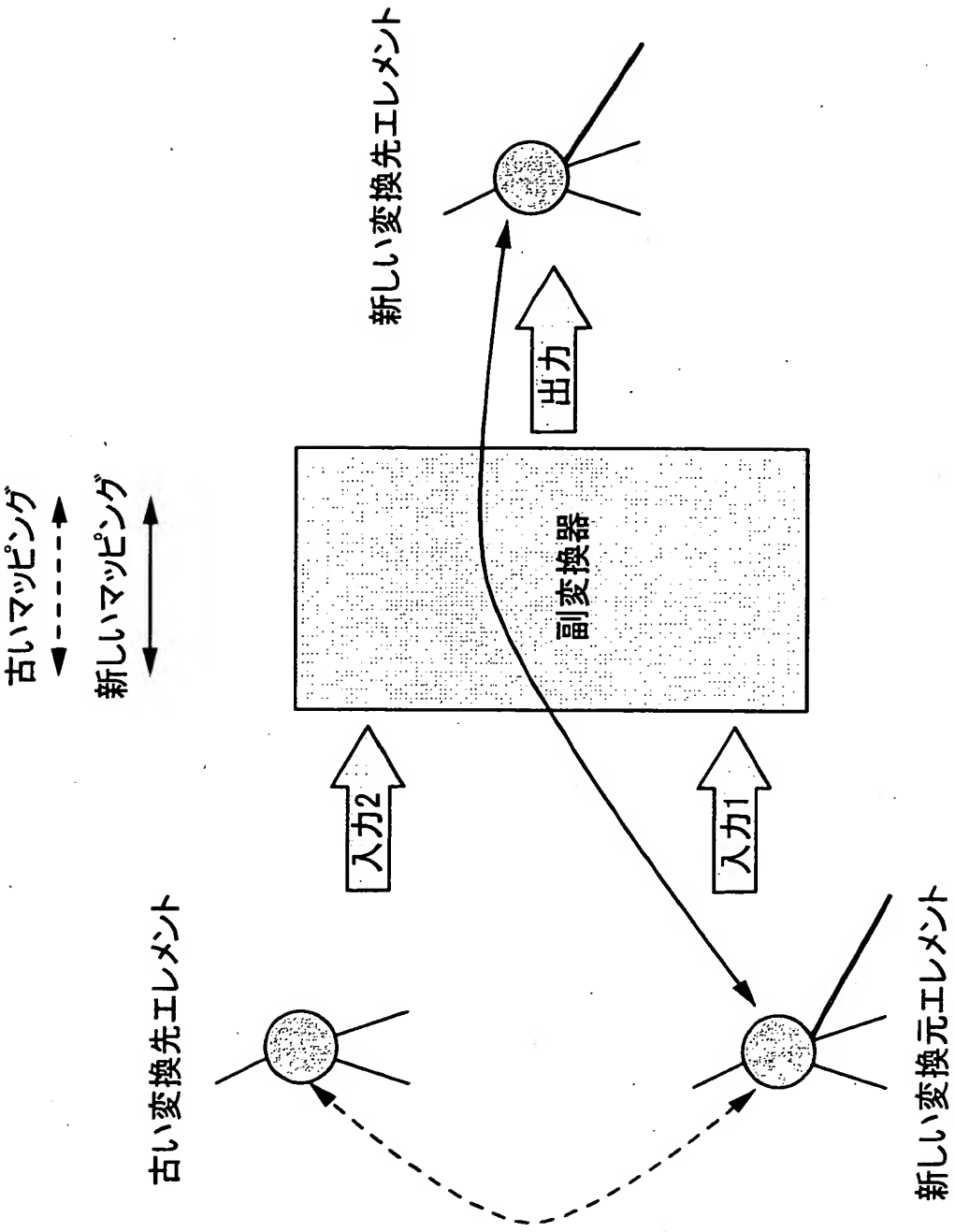
【図６】


アプリケーションA
モデルA
ビューA
アプリケーション
A実行部
10
モデルA
モデル
変換器
副変換器
40
41
モデルB
アプリケーション
B実行部
差分抽出部
イベント
生成部
31
32
30
アプリケーションB
モデルB
ビューB

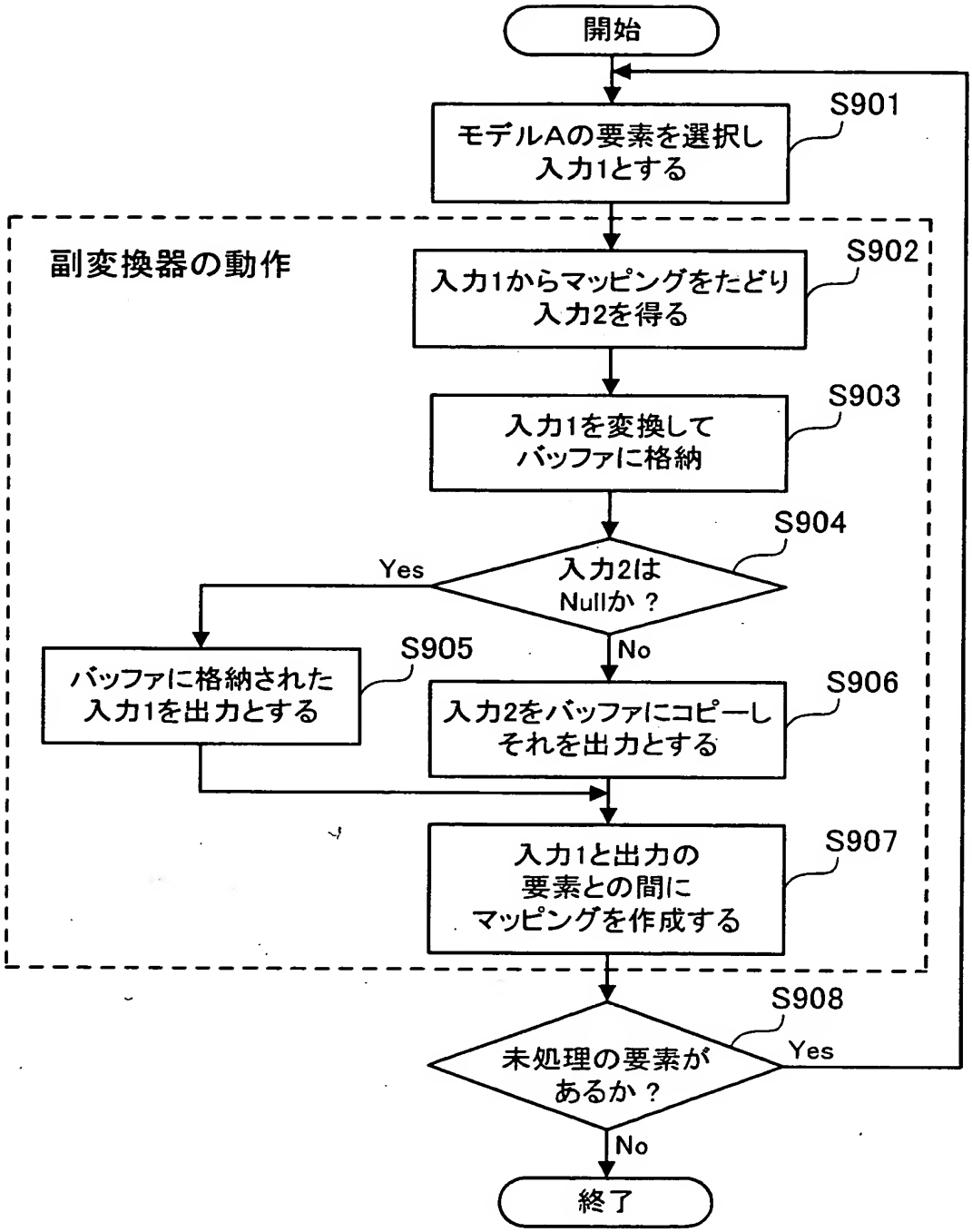
【図７】


古いマッピング
新しいマッピング
古い変換先モデル
新しい変換先モデル
新しく生成されるオブジェクト
再利用されるオブジェクト
入力2
入力1
出力
変換器
副変換器
新しい変換元モデル

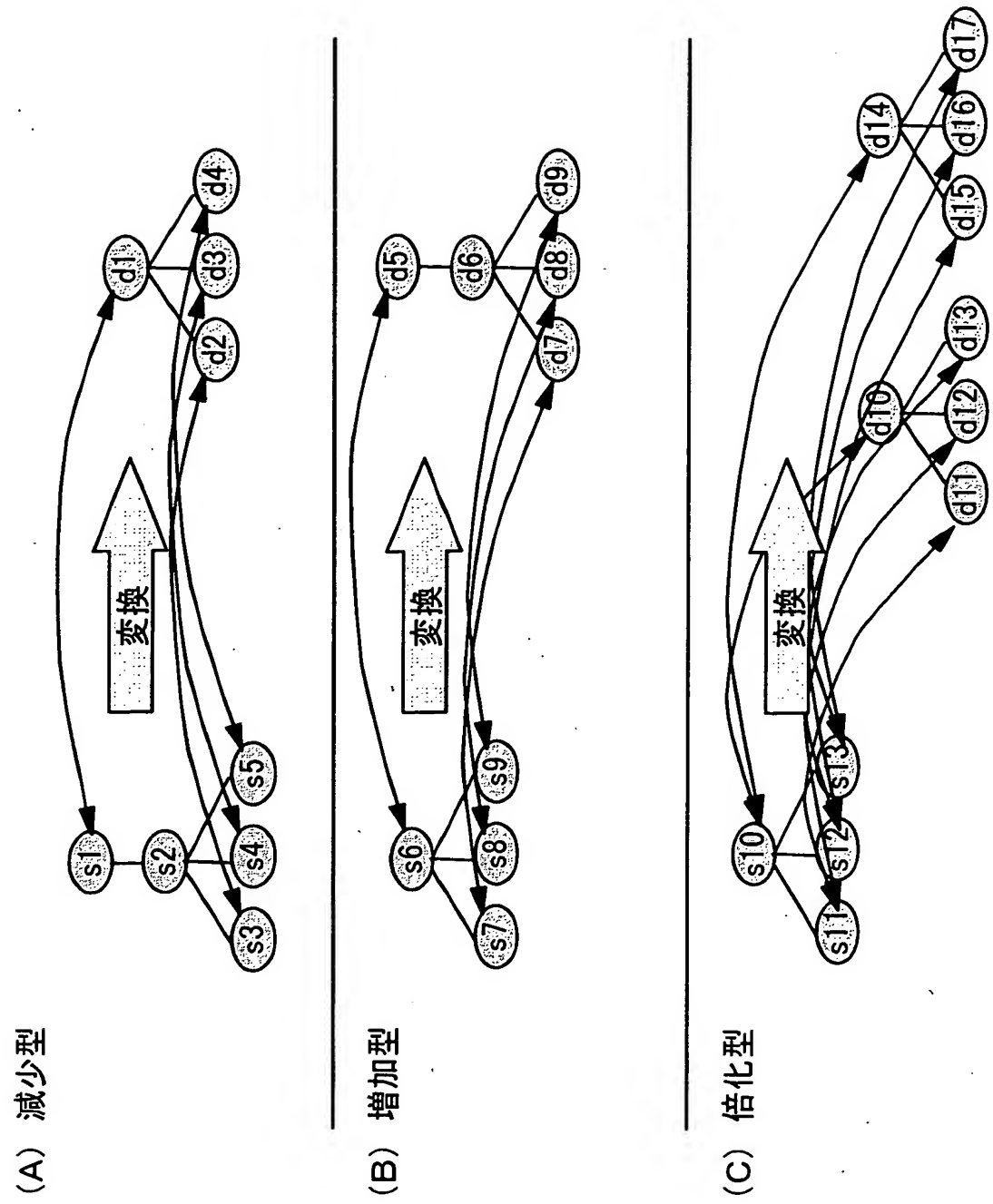
【図８】


新しい変換先エレメント
出力
副変換器
古いマッピング
新しいマッピング
入力2
入力1
古い変換先エレメント
新しい変換元エレメント

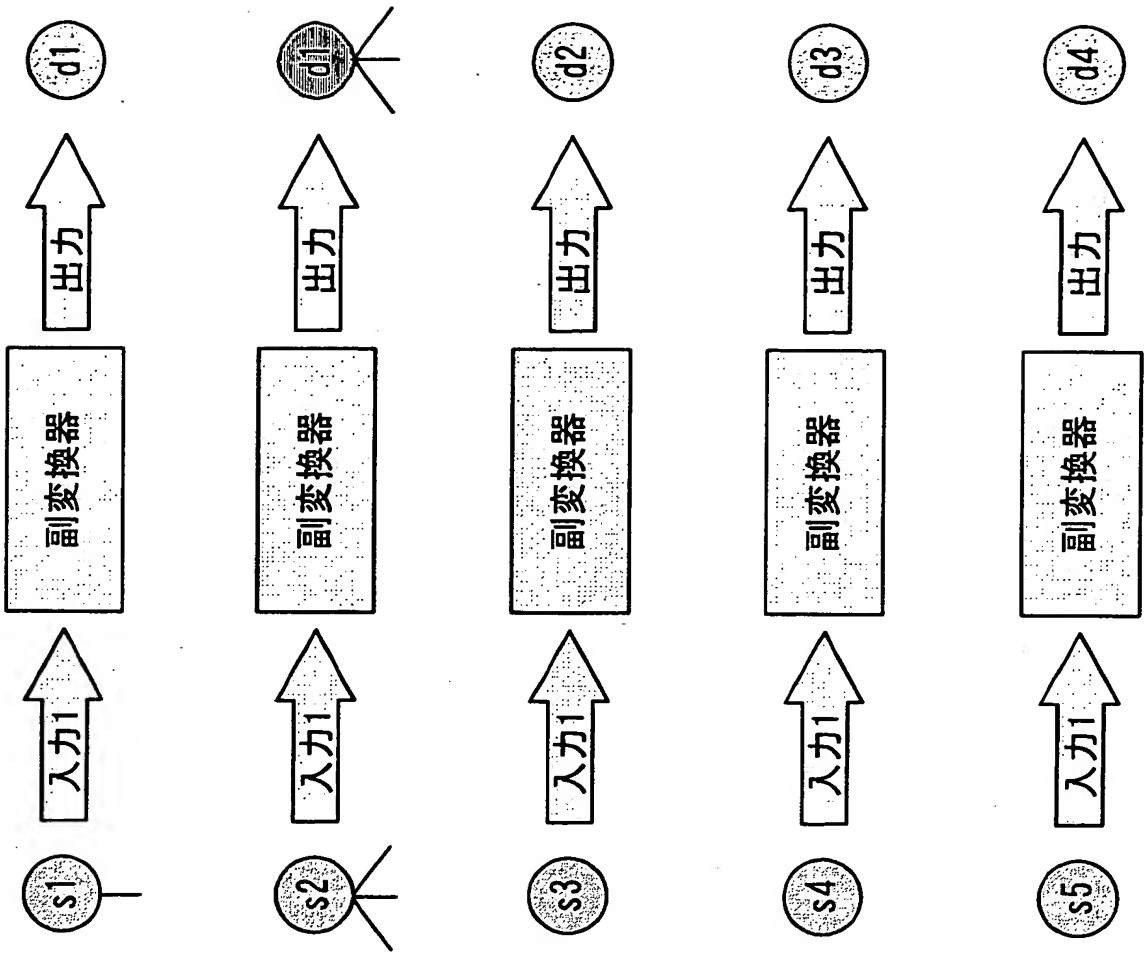
【図９】


開始
S901
モデルAの要素を選択し
入力1とする
副変換器の動作
S902
入力1からマッピングをたどり
入力2を得る
S903
入力1を変換して
バッファに格納
S904
入力2は
Nullか？
Yes
No
S905
バッファに格納された
入力1を出力とする
S906
入力2をバッファにコピーし
それを出力とする
S907
入力1と出力の
要素との間に
マッピングを作成する
S908
未処理の要素が
あるか？
Yes
No
終了

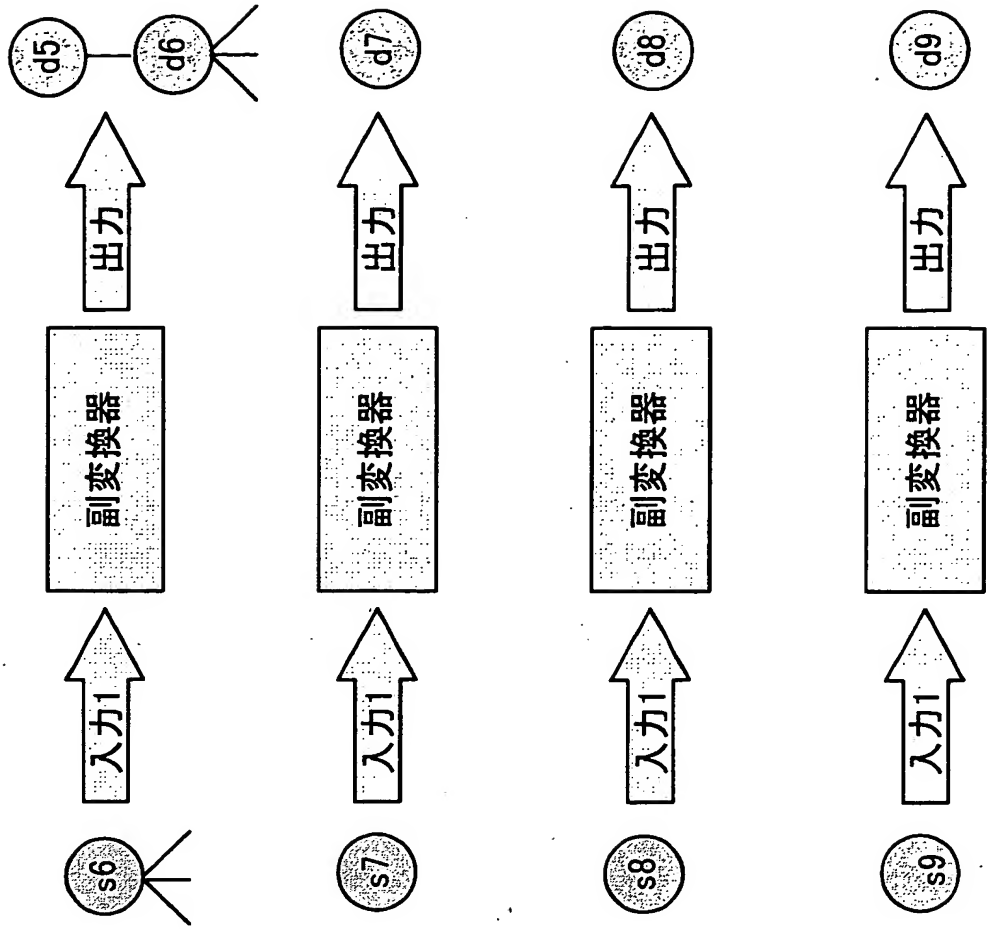
【図１０】


(A) 減少型
s1
s2
s3
s4
s5
d1
d2
d3
d4
変換
(B) 増加型
s6
s7
s8
s9
d5
d6
d7
d8
d9
変換
(C) 倍化型
s10
s11
s12
s13
d10
d11
d12
d13
d14
d15
d16
d17
変換

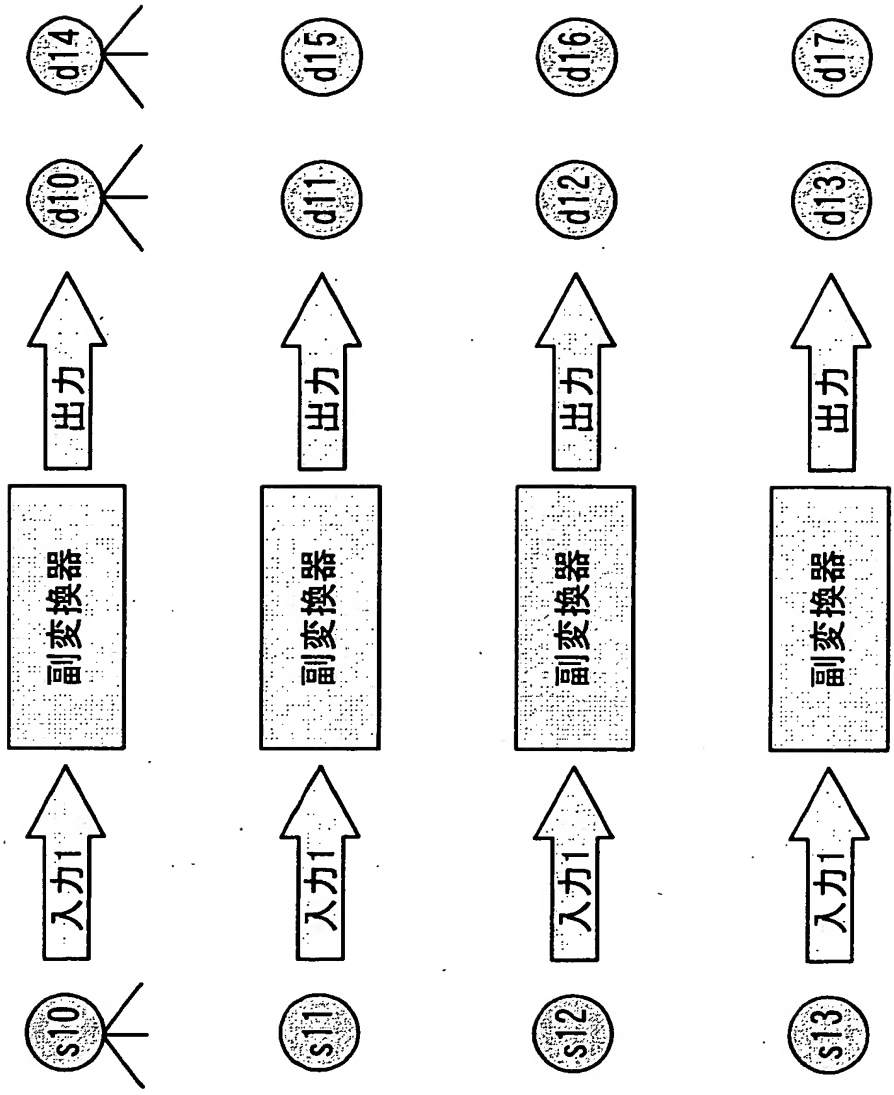
【図１１】


d1
出力
副変換器
入力1
s1
d1
出力
副変換器
入力1
s2
d2
出力
副変換器
入力1
s3
d3
出力
副変換器
入力1
s4
d4
出力
副変換器
入力1
s5

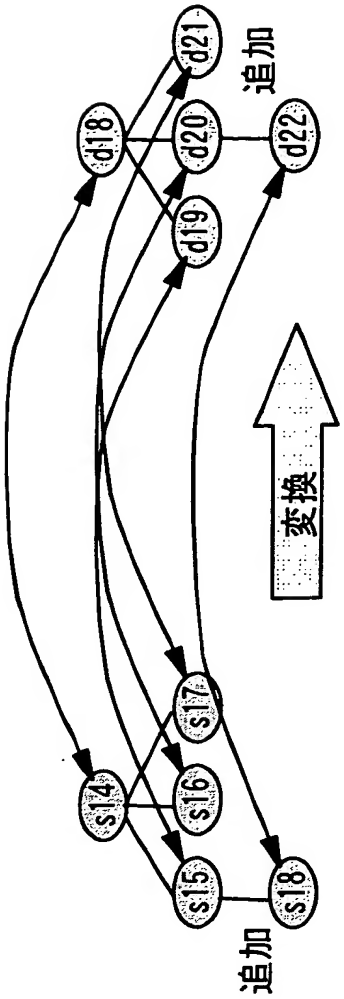
【図１２】


d5
d6
d7
d8
d9
出力
出力
出力
出力
副変換器
副変換器
副変換器
副変換器
入力1
入力1
入力1
入力1
s6
s7
s8
s9

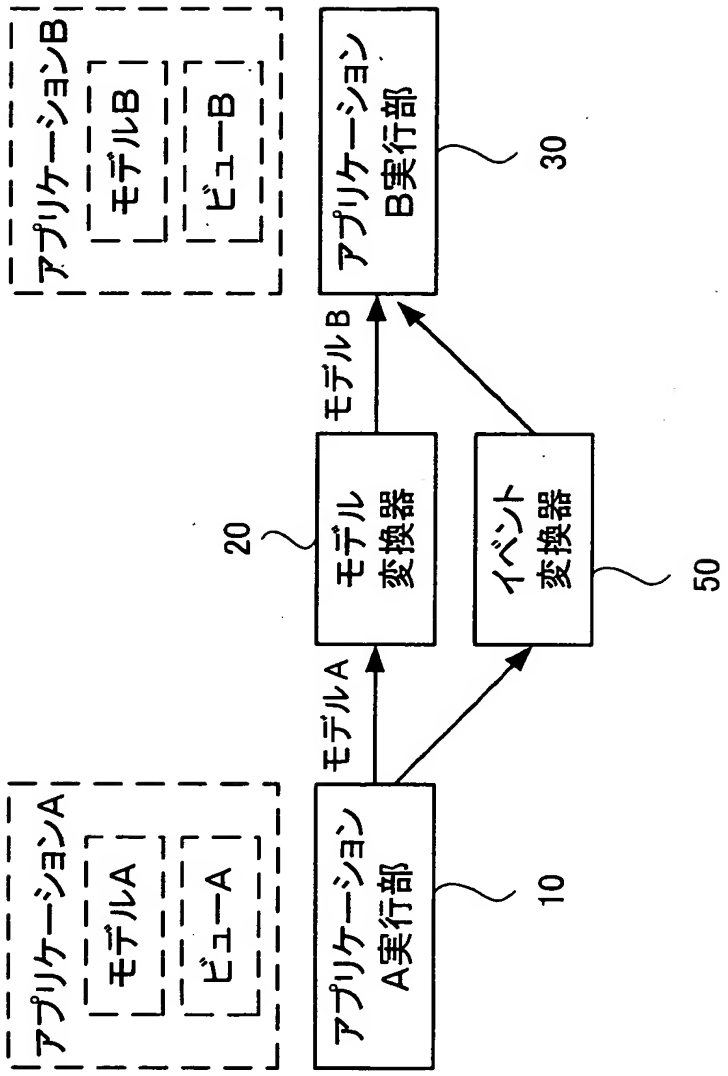
【図１３】



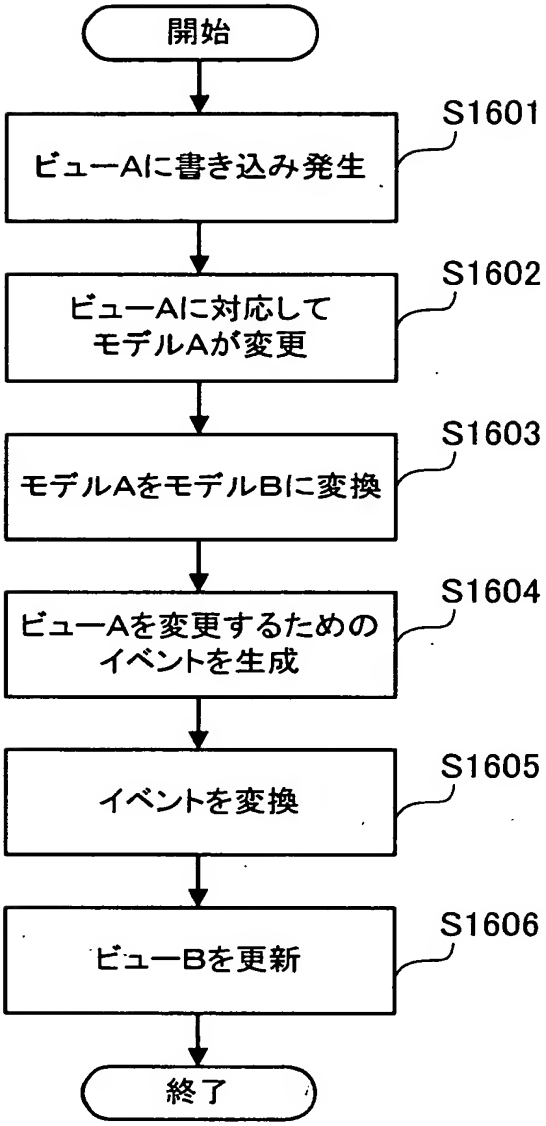
【図１４】



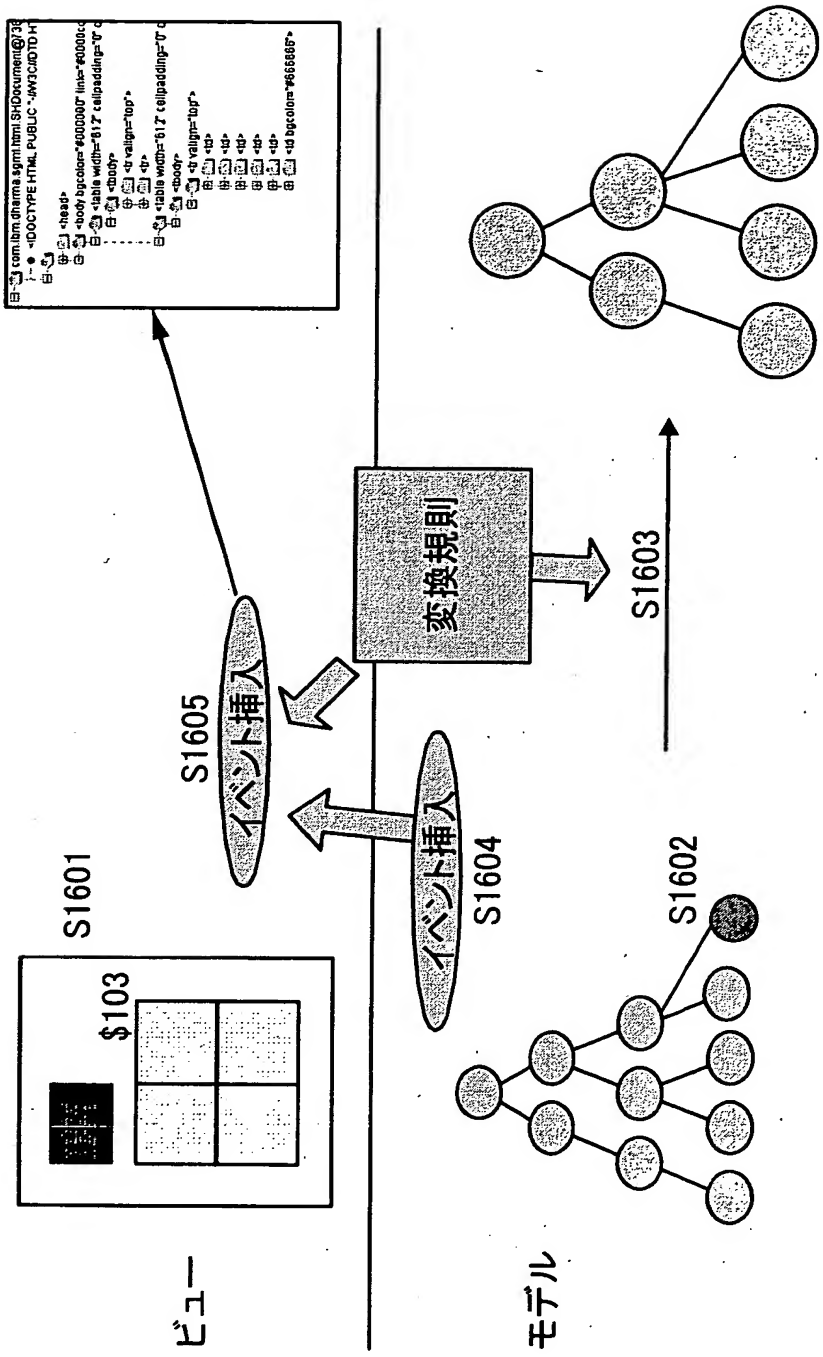
【図１５】


アプリケーションA
モデルA
ビューA
アプリケーション
A実行部
10
モデルA
モデル
変換器
20
モデルB
アプリケーション
B実行部
30
イベント
変換器
50
アプリケーションB
モデルB
ビューB

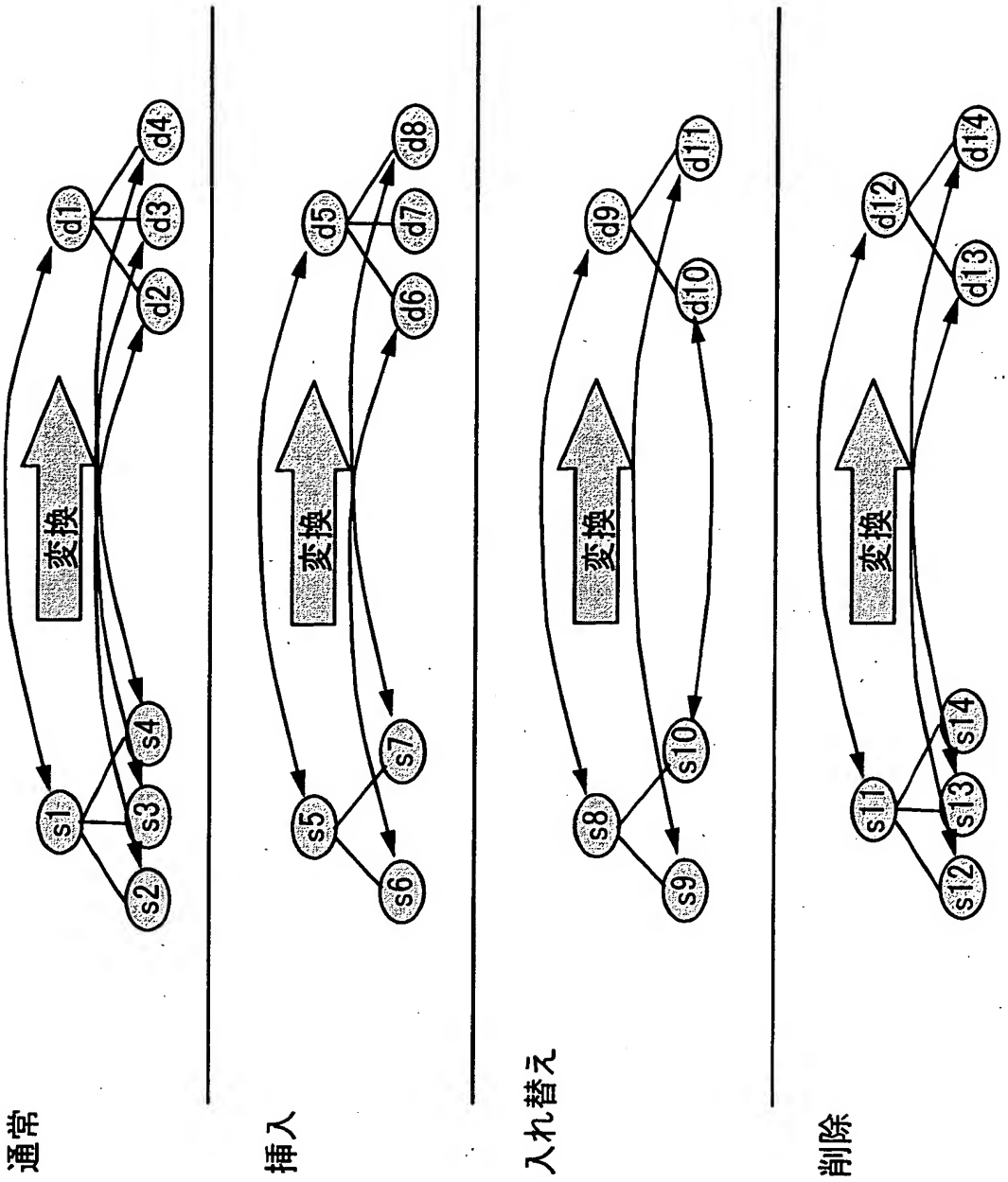
【図１６】


開始
S1601
ビューAに書き込み発生
S1602
ビューAに対応して
モデルAが変更
S1603
モデルAをモデルBに変換
S1604
ビューAを変更するための
イベントを生成
S1605
イベントを変換
S1606
ビューBを更新
終了

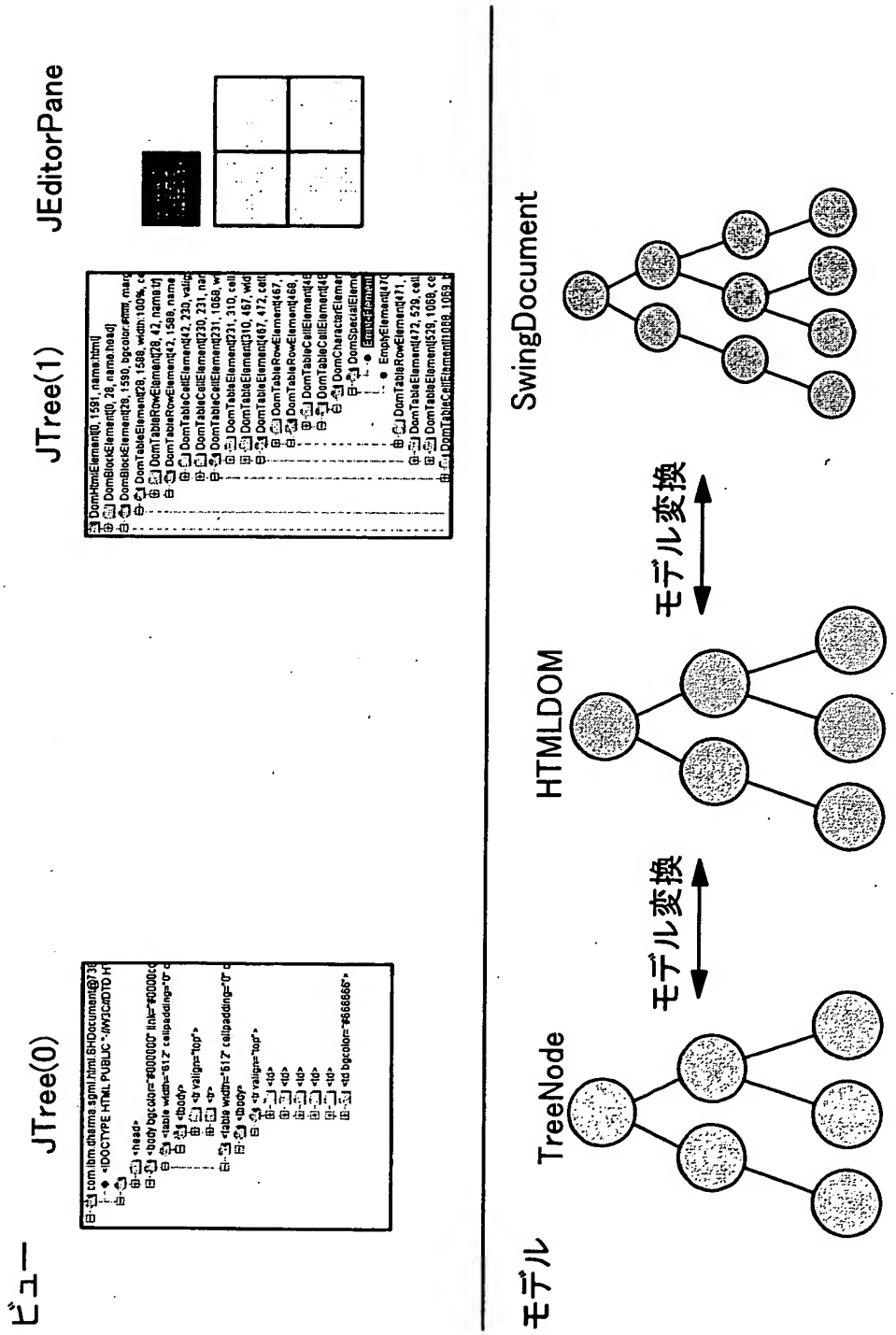
【図１７】


S1606
S1603
変換規則
S1605
イベント挿入
S1604
イベント挿入
S1601
S1602
$103
ビュー
モデル

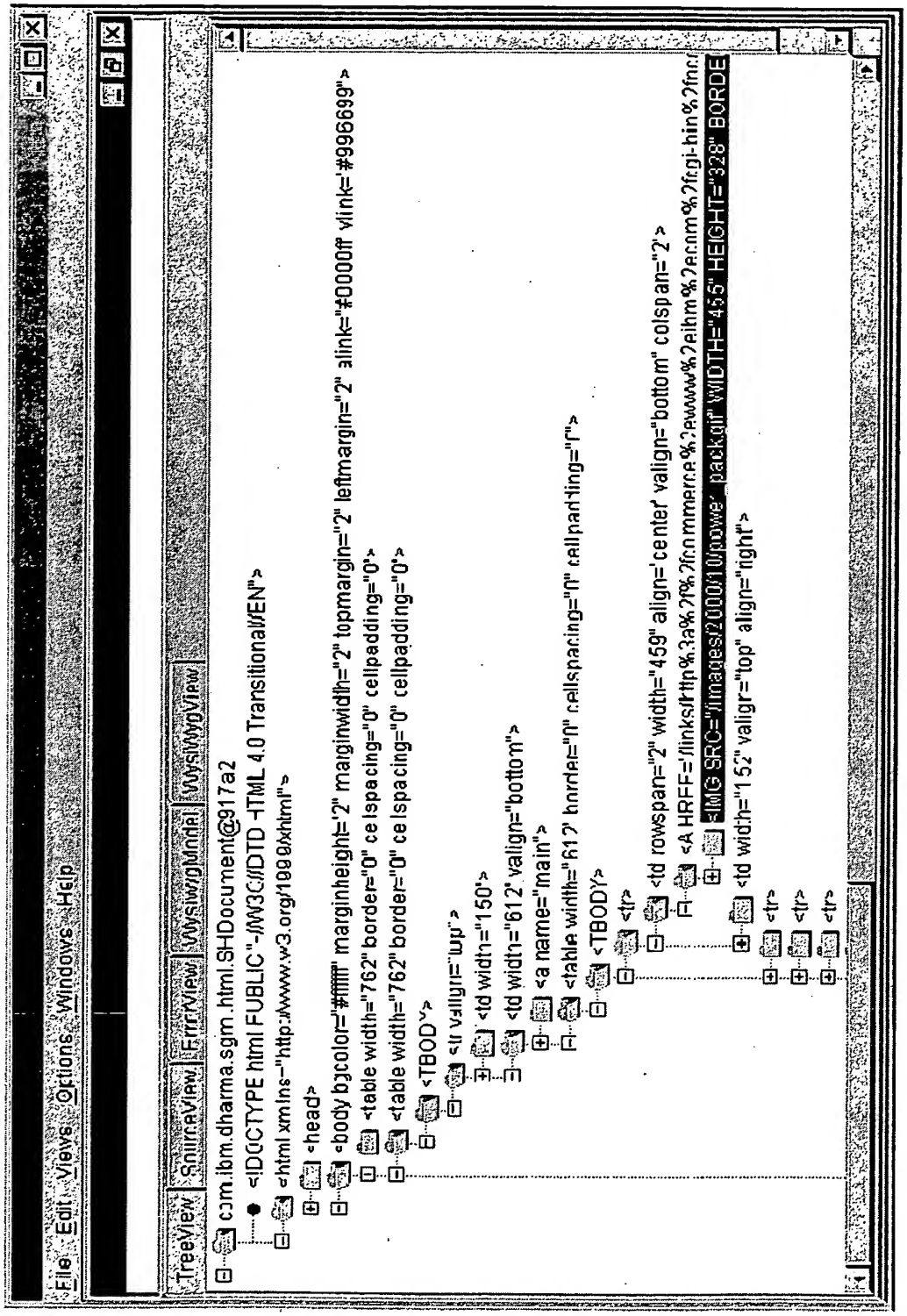
【図１８】


通常
変換
s1
s2
s3
s4
d1
d2
d3
d4
挿入
変換
s5
s6
s7
d5
d6
d7
d8
入れ替え
変換
s8
s9
s10
d9
d10
d11
削除
変換
s11
s12
s13
s14
d12
d13
d14

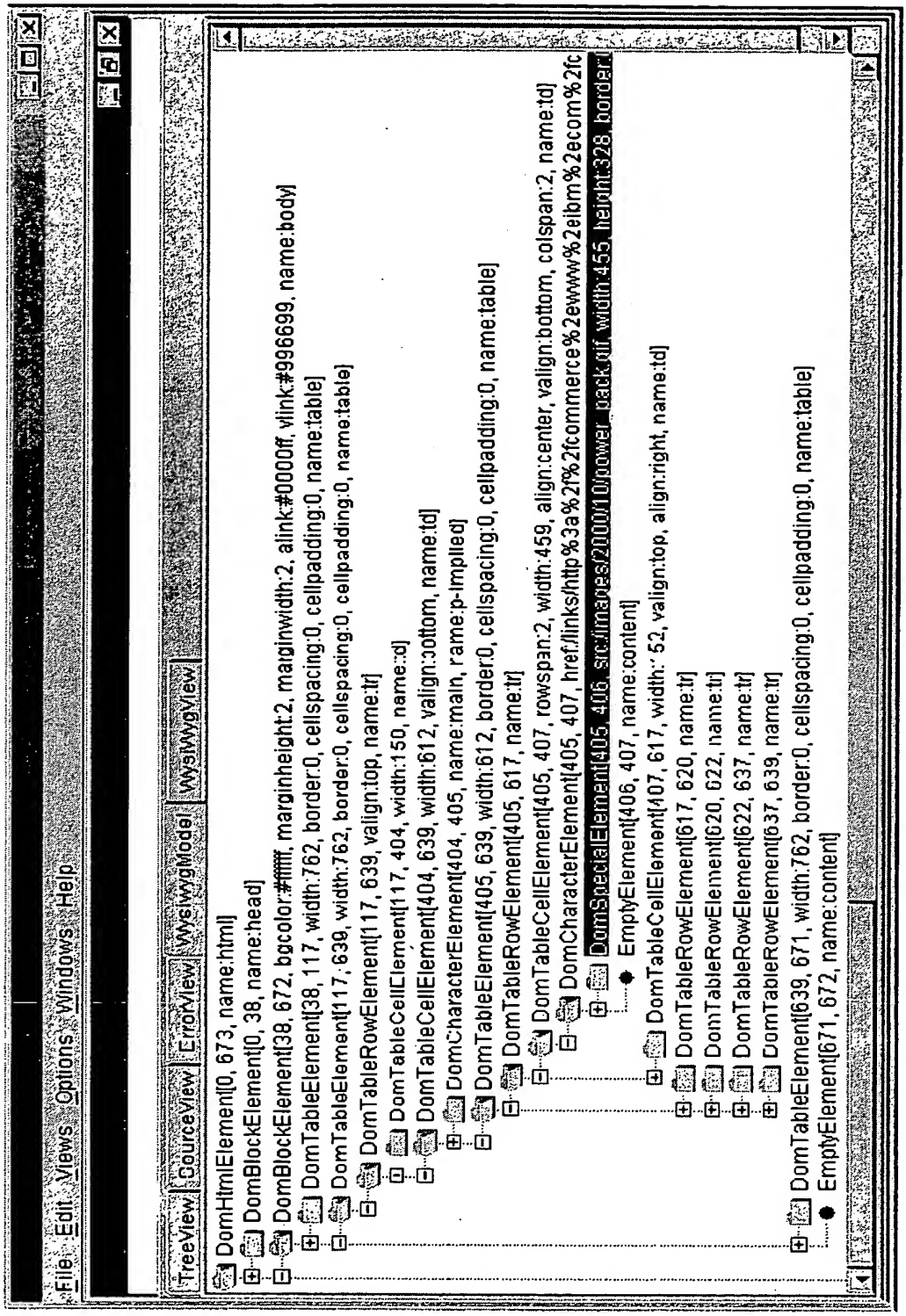
【図１９】


ビュー
JTree(0)
JTree(1)
JEditorPane
モデル
TreeNode
モデル変換
HTMLDOM
モデル変換
SwingDocument

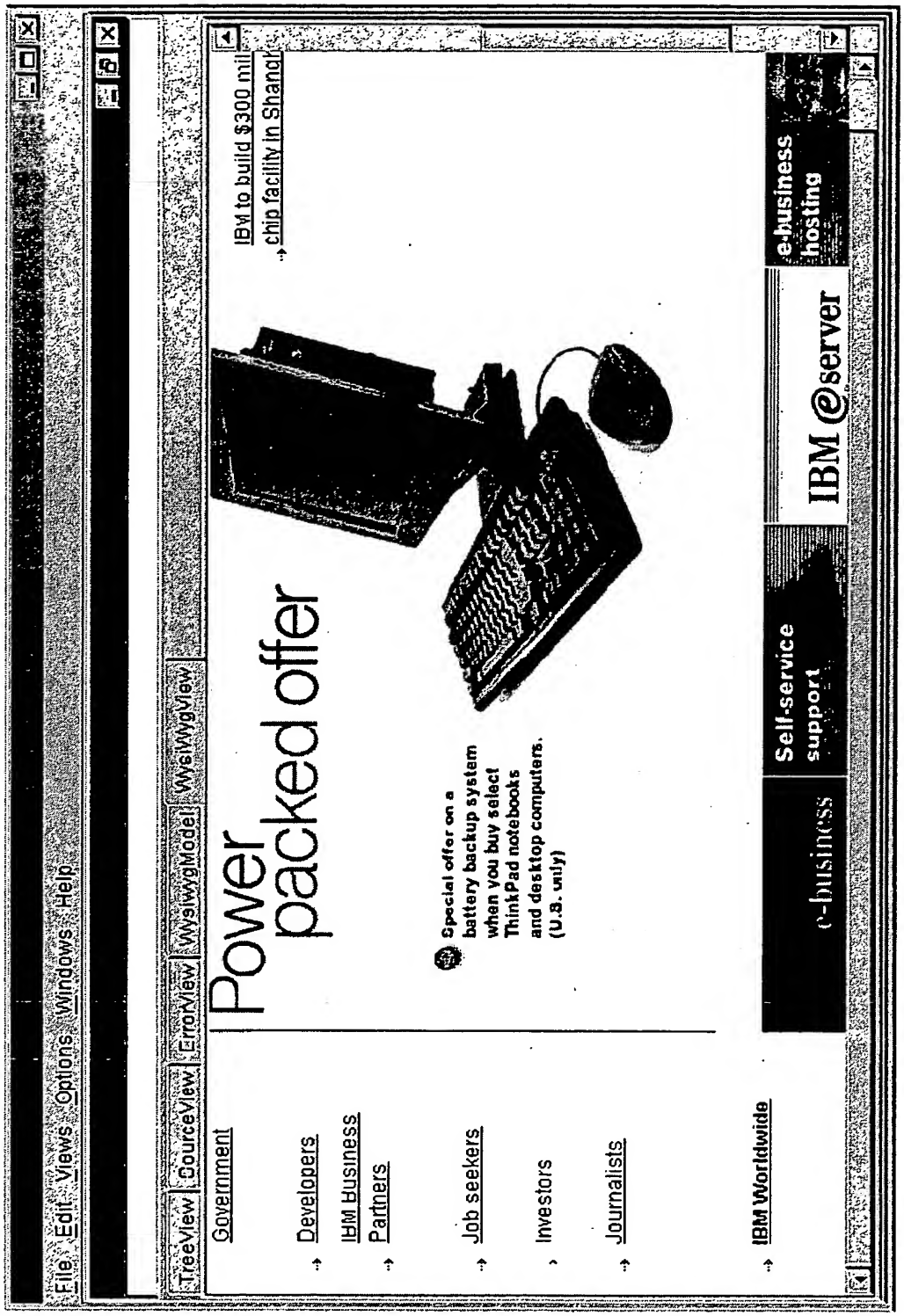
【図２０】


File Edit Views Options Windows Help
TreeView SnippetView ErrorView WysiwygModal WysiwygView
com.ibm.dharma.sgm.html.SHDocument@917a2
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<body bgcolor="#ffffff" marginheight="2" marginwidth="2" topmargin="2" leftmargin="2" alink="#0000ff" vlink="#996699">
<table width="762" border="0" cellspacing="0" cellpadding="0">
<table width="762" border="0" cellspacing="0" cellpadding="0">
<TBODY>
<tr valign="top">
<td width="150">
<td width="612" valign="bottom">
<a name="main">
<table width="612" border="0" cellspacing="0" cellpadding="0">
<TBODY>
<tr>
<td rowspan="2" width="459" align="center" valign="bottom" colspan="2">
<A HREF="/links/http%3a%2f%2fcommerce%2ewww%2eibm%2ecom%2fcgi-bin%2fmr
<IMG SRC="/images/2000/10/powe_pack.gif" WIDTH="455" HEIGHT="328" BORDE
<td width="152" valign="top" align="right">
<tr>
<tr>
<tr>

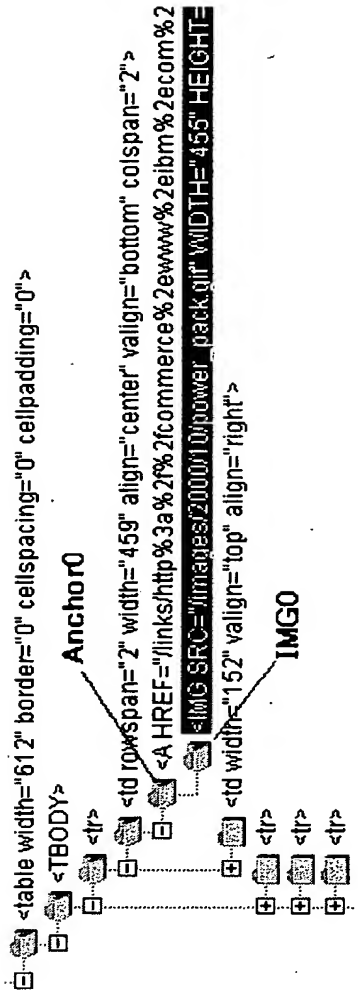
【図２１】


File Edit Views Options Windows Help
TreeView SourceView ErrorView WysiwygMode WysiWygView
DomHtmlElement[0, 673, name:html]
DomBlockElement[0, 38, name:head]
DomBlockElement[38, 672, bgcolor:#ffffff, marginheight:2, marginwidth:2, alink:#0000ff, vlink:#996699, name:body]
DomTableElement[38, 117, width:762, border:0, cellspacing:0, cellpadding:0, name:table]
DomTableElement[117, 639, width:762, border:0, cellspacing:0, cellpadding:0, name:table]
DomTableRowElement[117, 639, valign:top, name:tr]
DomTableCellElement[117, 404, width:150, name:td]
DomTableCellElement[404, 639, width:612, valign:bottom, name:td]
DomCharacterElement[404, 405, name:main, name:p-implied]
DomTableElement[405, 639, width:612, border:0, cellspacing:0, cellpadding:0, name:table]
DomTableRowElement[405, 617, name:tr]
DomTableCellElement[405, 407, rowspan:2, width:459, align:center, valign:bottom, colspan:2, name:td]
DomCharacterElement[405, 407, href:/links/http%3a%2f%2fcommerce%2ewww%2eibm%2ecom%2fc
DomSpecialElement[405, 406, src:/images/2000l10/power_pack.gif, width:455, height:328, border
EmptyElement[406, 407, name:content]
DomTableCellElement[407, 617, width:52, valign:top, align:right, name:td]
DomTableRowElement[617, 620, name:tr]
DomTableRowElement[620, 622, name:tr]
DomTableRowElement[622, 637, name:tr]
DomTableRowElement[637, 639, name:tr]
DomTableElement[639, 671, width:762, border:0, cellspacing:0, cellpadding:0, name:table]
EmptyElement[671, 672, name:content]

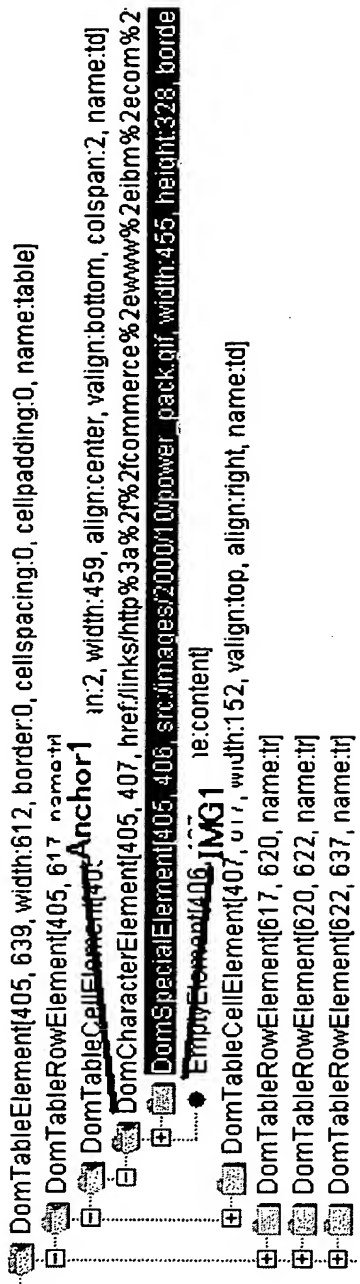
【図２２】


File Edit Views Options Windows Help
TreeView CourceView ErrorView WysiwygModel WysiWygView
Government
Developers
IBM business Partners
Job seekers
Investors
Journalists
IBM Worldwide
Power packed offer
Special offer on a battery backup system when you buy select ThinkPad notebooks and desktop computers. (U.S. only)
IBM to build $300 mil chip facility in Shangh
e-business
Self-service support
IBM @server
e-business hosting

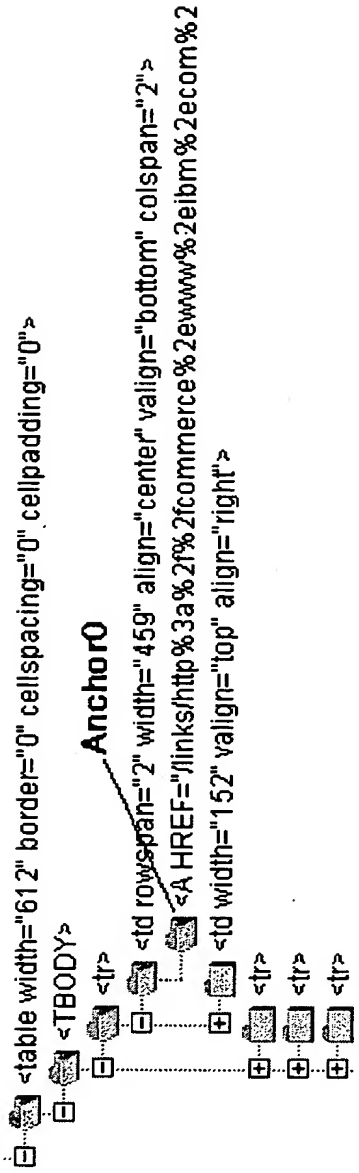
【図２３】


<table width="612" border="0" cellspacing="0" cellpadding="0">
<TBODY>
<tr>
<td rowspan="2" width="459" align="center" valign="bottom" colspan="2">
<A HREF="/links/http%3a%2f%2fcommerce%2ewww%2eibm%2ecom%2
<IMG SRC="/images/2000010/power_pack.gif" WIDTH="455" HEIGHT=
<td width="152" valign="top" align="right">
<tr>
<tr>
<tr>
Anchor0
IMG0

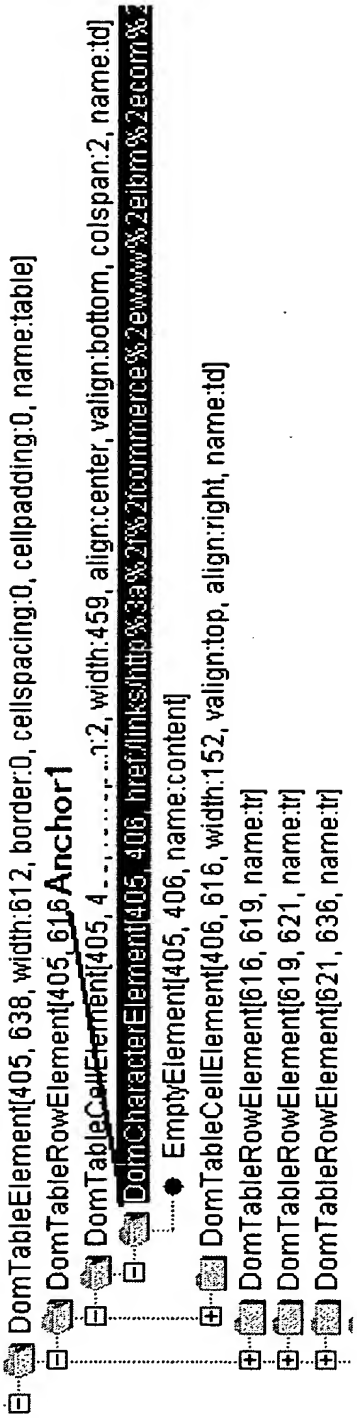
【図２４】


DomTableElement[405, 639, width:612, border:0, cellspacing:0, cellpadding:0, name:table]
DomTableRowElement[405, 617, name:tr]
DomTableCellElement[405
Anchor1
in:2, width:459, align:center, valign:bottom, colspan:2, name:td]
DomCharacterElement[405, 407, href:/links/http%3a%2f%2fcommerce%2ewww%2eibm%2ecom%2
DomSpecialElement[405, 406, src:/images/2000/10/power_pack.gif, width:455, height:328, borde
EmptyElement[406
IMG1
le:content]
DomTableCellElement[407, 617, width:152, valign:top, align:right, name:td]
DomTableRowElement[617, 620, name:tr]
DomTableRowElement[620, 622, name:tr]
DomTableRowElement[622, 637, name:tr]

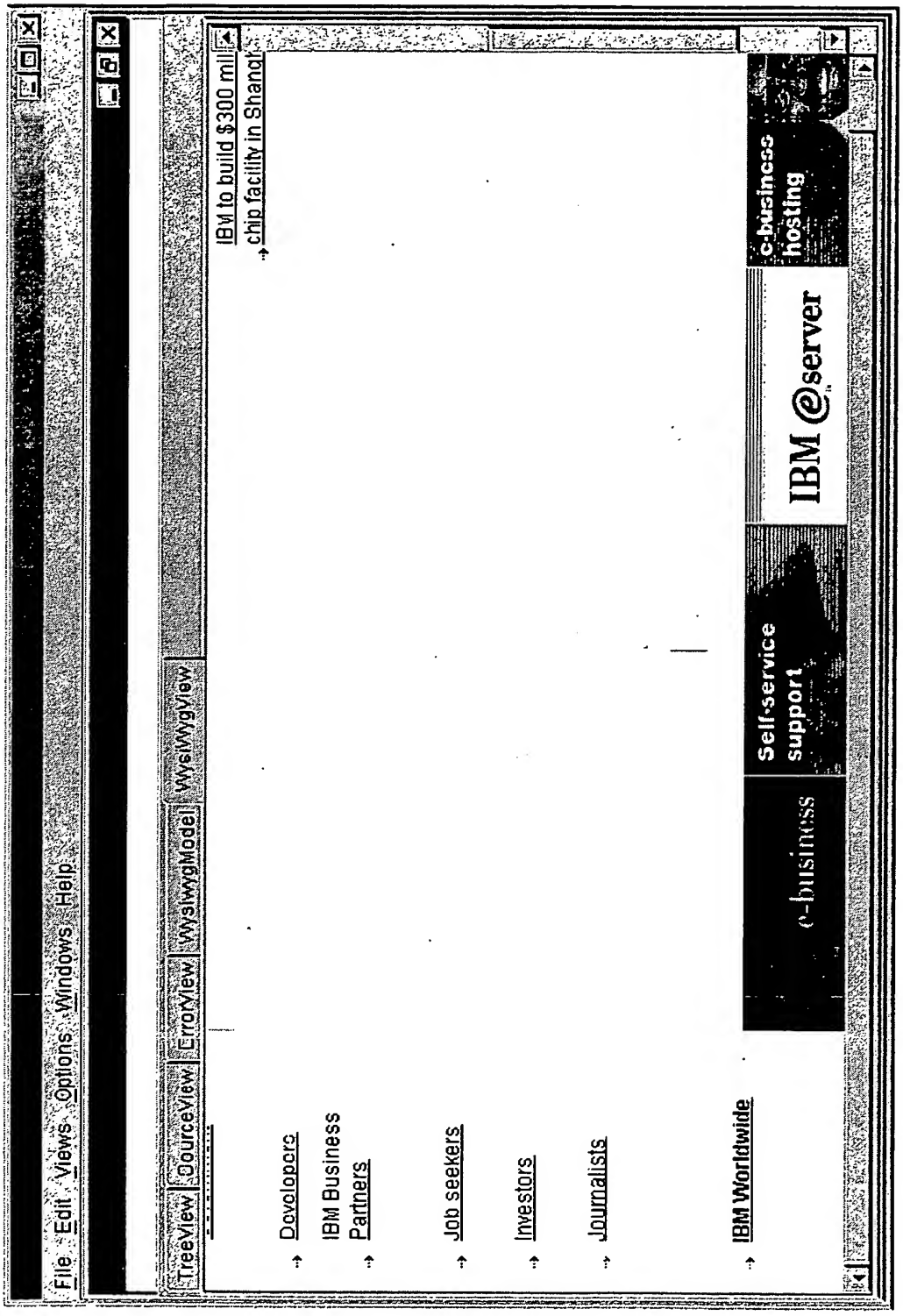
【図２５】


<table width="612" border="0" cellspacing="0" cellpadding="0">
<TBODY>
<tr>
<td rowspan="2" width="459" align="center" valign="bottom" colspan="2">
Anchor0
<A HREF="/links/http%3a%2f%2fcommerce%2ewww%2eibm%2ecom%2
<td width="152" valign="top" align="right">
<tr>
<tr>
<tr>

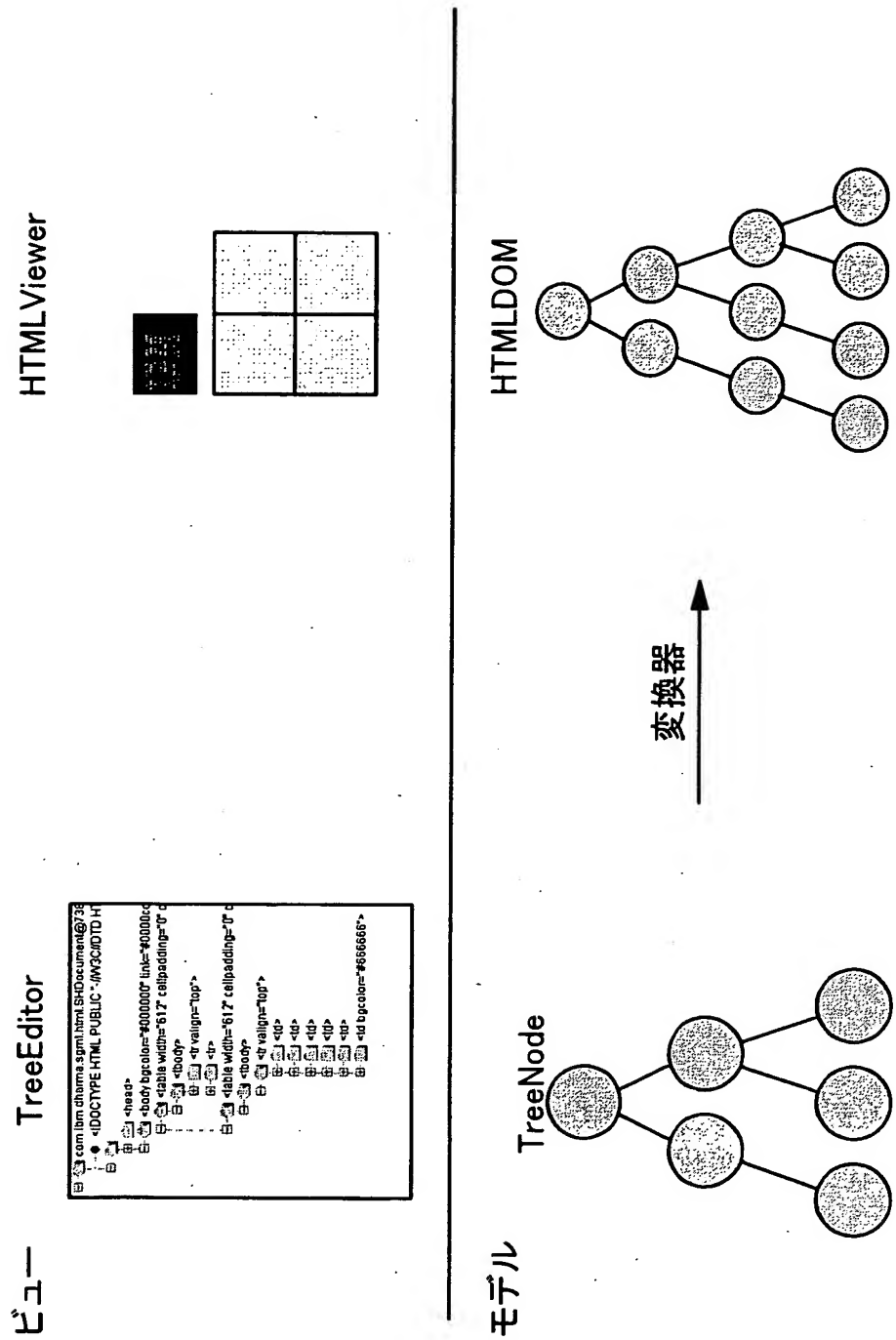
【図２６】


DomTableElement[405, 638, width:612, border:0, cellspacing:0, cellpadding:0, name:table]
DomTableRowElement[405, 616 Anchor1
DomTableCellElement[405, 4
:1:2, width:459, align:center, valign:bottom, colspan:2, name:td]
DomCharacterElement[405, 406, href://links/http%3a%2f%2fcommerce%2ewww%2eibm%2ecom%2
EmptyElement[405, 406, name:content]
DomTableCellElement[406, 616, width:152, valign:top, align:right, name:td]
DomTableRowElement[616, 619, name:tr]
DomTableRowElement[619, 621, name:tr]
DomTableRowElement[621, 636, name:tr]

【図２７】


File Edit Views Options Windows Help
TreeView SourceView ErrorView WysiwygModel WysiWygView
IBM to build $300 mil
chip facility in Shangt
Developers
IBM Business Partners
Job seekers
Investors
Journalists
IBM Worldwide
e-business
Self-service support
IBM @server
e-business hosting

【図２８】



【図２９】

```
<?xml version="1.0"?>
<!ELEMENT discussion thread*>
<!ATTLIST discussion name CDATA #REQUIRED>
<!ELEMENT thread statement+>
<!ATTLIST thread id ID #IMPLIED>
<!ELEMENT statement (content, statement*)>
<!ATTLIST statement
  subject CDATA ""
  writtenby CDATA "anonymous"
  id ID #IMPLIED
>
<!ELEMENT content #PCDATA>
```

【図３０】

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/TR/xhtml1/strict">
<xsl:template match="discussion">
 <html>
  <head>
   <title>
    <xsl:valueof select="@name"/>
   </title>
  </head>
  <body>
   <xsl:apply-templates/>
  </body>
 </html>
</xsl:template>
<xsl:template match="thread">
 <ul>
  <xsl:apply-templates/>
 </ul>
</xsl:template>
<xsl:template match="statement">
 <li><b><xsl:valueof select="@subject"/></b> - <i><xsl:valueof
select="@writtenby"/></i><br/>
   <xsl:apply-templates select="content"/>
   <ul>
    <xsl:apply-templates select="statement"/>
   </ul>
  </li>
</xsl:template>
<xsl:template match="content">
 <xsl:apply-templates/>
</xsl:template>
```

【図３１】

```
<?xml encoding="US-ASCII"?>
<!DOCTYPE discussion SYSTEM "discussion.dtd">
<discussion name="Transformer Discussion">
 <thread id="1">
  <statement subject="Test" writtenby="kondo">
   <content>This is a test.</content>
  </statement>
 </thread>
</discussion>
```

【図３２】

```
<html>
<head>
<title>Transformer Discussion</title>
</head>
<body>
<ul>
<li><b>Test</b> - <i>kondo</i><br>
This is a test.
<ul>
</ul>
</ul>
</body>
</html>
```

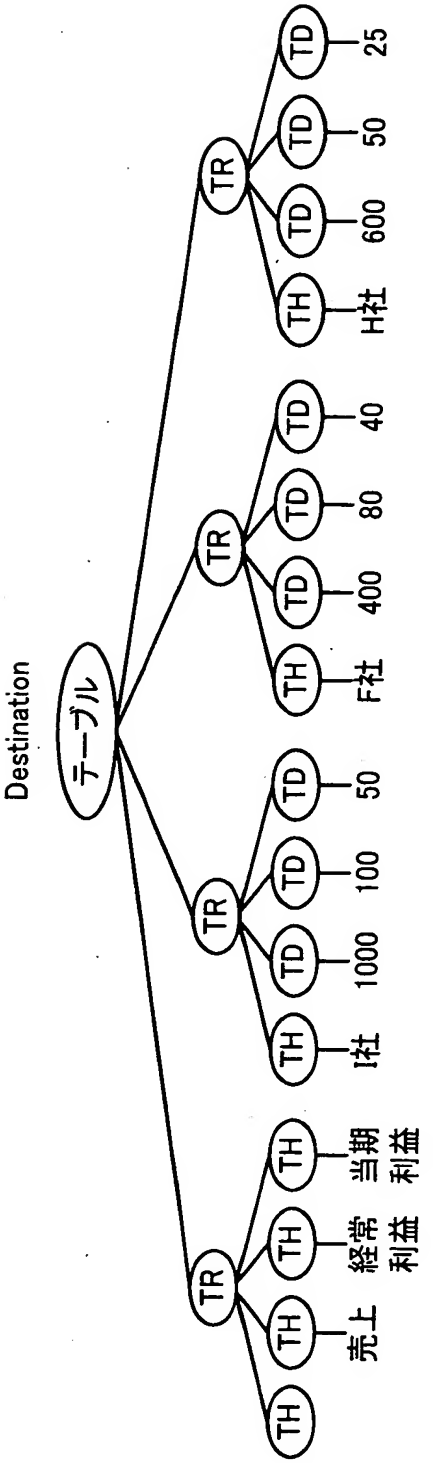
【図３３】

```
<html>
<head>
<title>Transformer Discussion</title>
</head>
<body>
<ul>
<li><b>Test</b> - <i>kondo</i><br>
This is a test.
<ul>
<li><b>OK</b> - <i>anonymous</i><br>
OK
</ul>
</ul>
</body>
</html>
```

【図３４】

ソース

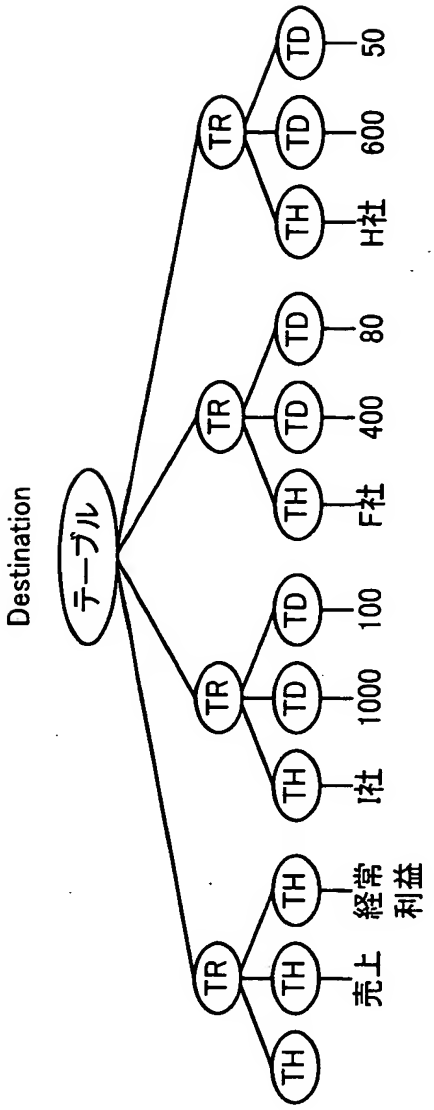
	売上	経常利益	当期利益
I社	1000	100	50
F社	400	80	40
H社	600	50	25



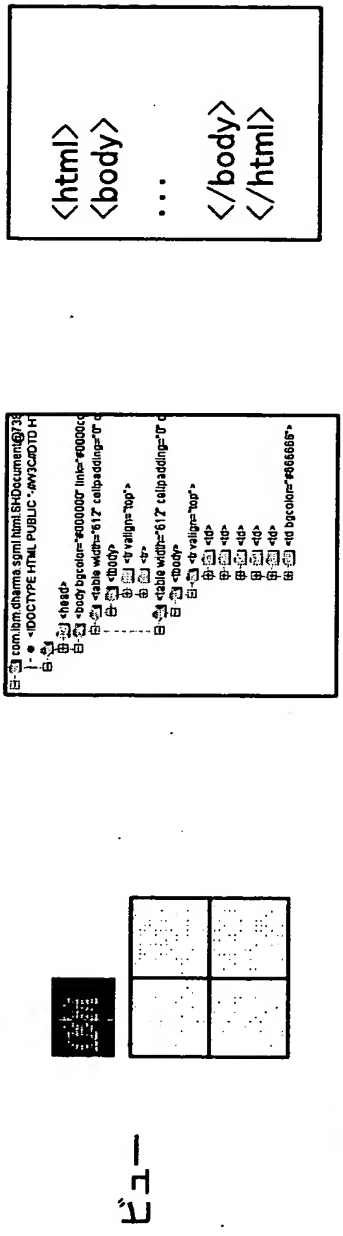
【図３５】

ソース

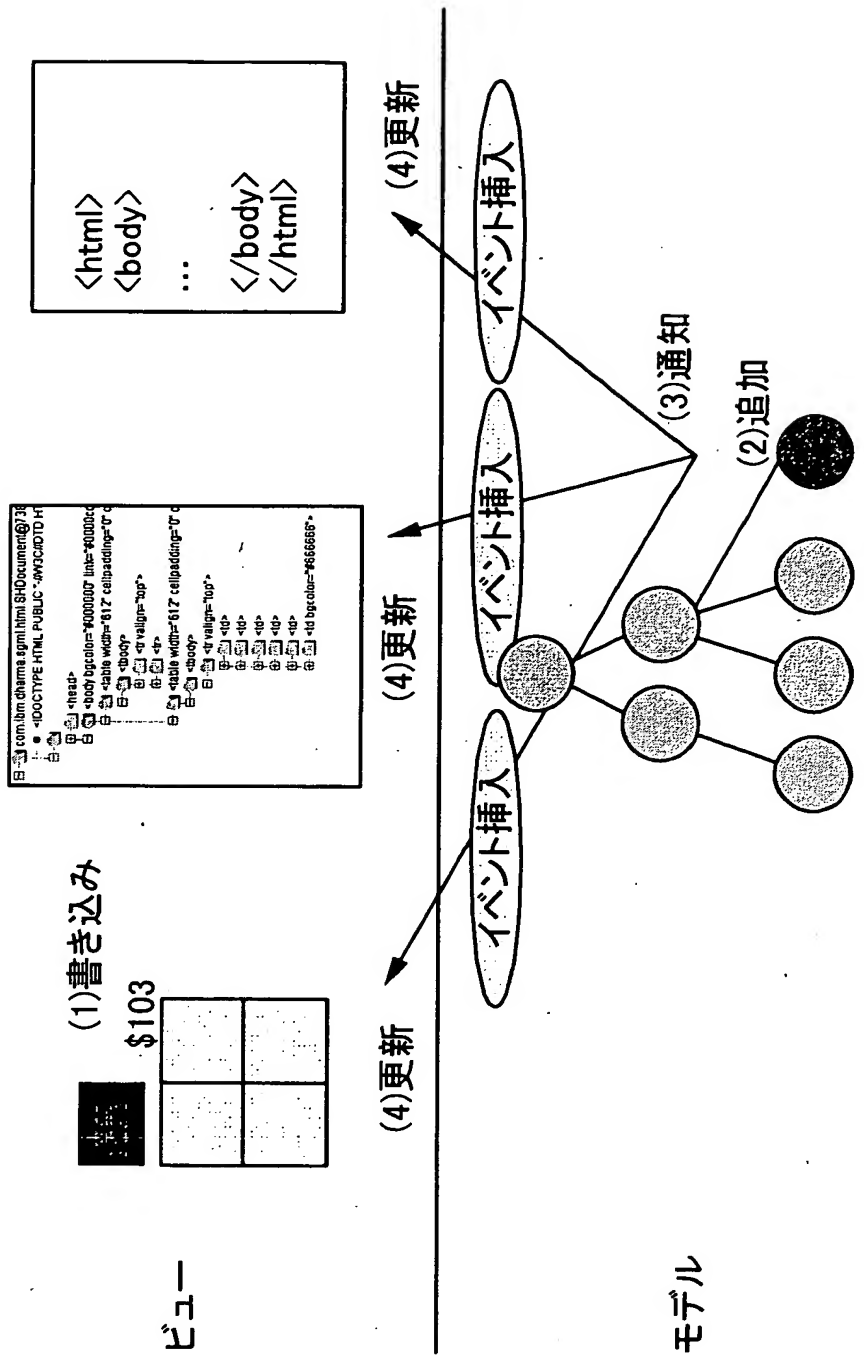
	売上	経常利益
I社	1000	100
F社	400	80
H社	600	50



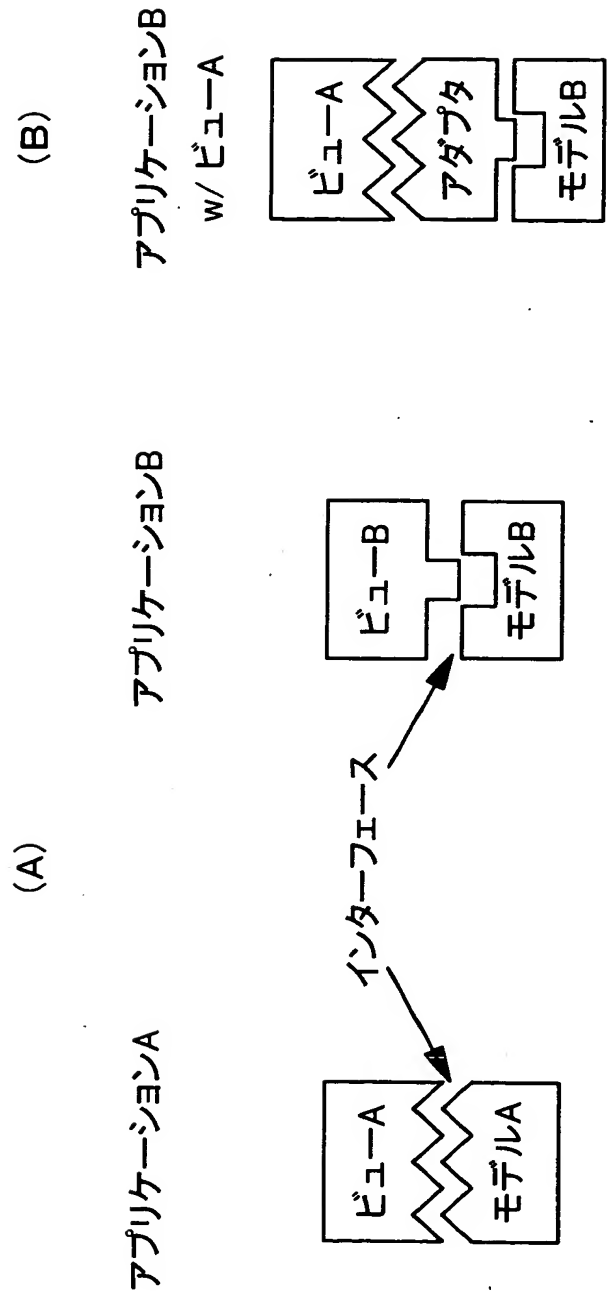
【図３６】


<html>
<body>
...
</body>
</html>
com.ibm.dharma.sgml.html.SHDocument@73
<!DOCTYPE HTML PUBLIC "-//W3C//DTD H
<head>
<body bgcolor="#000000" link="#00000
<table width="617" cellpadding="0"
<tbody>
<tr valign="top">
<td>
<table width="617" cellpadding="0"
<tbody>
<tr valign="top">
<td>
<td>
<td>
<td>
<td>
<td bgcolor="#666666">
ビュー
モデル

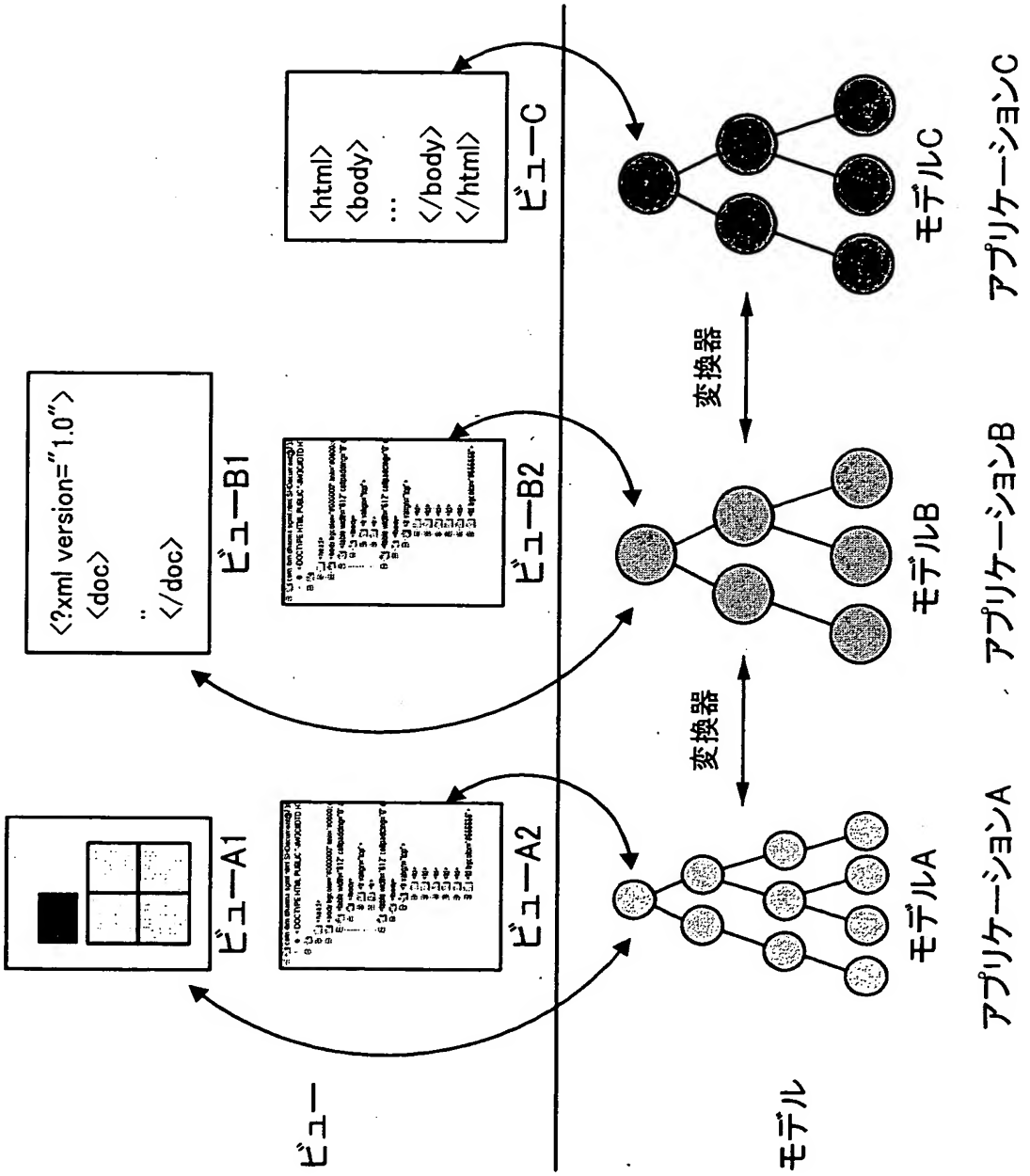
【図３７】


ビュー
モデル
(1)書き込み
$103
(4)更新
(4)更新
(4)更新
イベント挿入
イベント挿入
イベント挿入
(3)通知
(2)追加
<html>
<body>
...
</body>
</html>

【図３８】


(A)
アプリケーションA
ビューA
モデルA
インターフェース
アプリケーションB
ビューB
モデルB
(B)
アプリケーションB
w/ ビューA
ビューA
アダプタ
モデルB

【図３９】


ビュー
モデル
ビューA1
ビューA2
ビューB1
ビューB2
ビューC
<?xml version="1.0"?>
<doc>
...
</doc>
<html>
<body>
...
</body>
</html>
変換器
変換器
モデルA
モデルB
モデルC
アプリケーションA
アプリケーションB
アプリケーションC

【書類名】　　要約書

【要約】

【課題】　モデル変換を用いて所定のモデルから他のアプリケーションのビューを使用する場合に、変換元のモデルに対する変更を動的に変換先のモデル及びビューに反映させ、ビューを更新できるようにする。

【解決手段】　モデルとビューとを別個に有するアプリケーションの編集を行うアプリケーション編集装置において、このアプリケーションにおける第１のモデルを編集するアプリケーションＡ実行部１０と、この第１のモデルを第２のモデルに変換するモデル変換器２０と、この第２のモデルをそのビューにてディスプレイ装置に表示するアプリケーションＢ実行部３０とを備える。そして、このアプリケーションＢ実行部３０は、第２のモデルが更新された場合に、この第２のモデルの更新に基づくイベントを生成するイベント生成部３２を備え、このイベント生成部３２にて生成されたイベントに基づいてディスプレイ装置に表示されたビューを変更する。

【選択図】　　図２

認定・付加情報

特許出願の番号	特願２００１－２４６２９０
受付番号	５０１０１１９７１６７
書類名	特許願
担当官	風戸　勝利　　９０８３
作成日	平成１３年　９月２５日

＜認定情報・付加情報＞

【特許出願人】
- 【識別番号】 390009531
- 【住所又は居所】 アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
- 【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】
- 【識別番号】 100086243
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 坂口　博

【代理人】
- 【識別番号】 100091568
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内
- 【氏名又は名称】 市位　嘉宏

【代理人】
- 【識別番号】 100106699
- 【住所又は居所】 神奈川県大和市下鶴間１６２３番１４　日本アイ・ビー・エム株式会社大和事業所内
- 【氏名又は名称】 渡部　弘道

【復代理人】 申請人
- 【識別番号】 100104880
- 【住所又は居所】 東京都港区赤坂５－４－１１　山口建設第２ビル６Ｆ　セリオ国際特許事務所
- 【氏名又は名称】 古部　次郎

【選任した復代理人】
- 【識別番号】 100100077

次頁有

認定・付加情報（続き）

【住所又は居所】　東京都港区赤坂５－４－１１　山口建設第２ビル
６Ｆ　セリオ国際特許事務所
【氏名又は名称】　大場　充

次頁無

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　２０００年　５月１６日
［変更理由］　名称変更
住　所　　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
氏　名　　インターナショナル・ビジネス・マシーンズ・コーポレーション